IFIP WG 7.3 Performance 2013

31st International Symposium on Computer Performance, Modeling, Measurements and Evaluation 2013

Student Poster Abstracts

September 24-26, Vienna, Austria

Editorial Message from the Student Poster Chair

The poster proceedings contain the extended abstracts of the accepted student posters of Performance 2013, held at the University of Vienna, Austria, on September 24th-26th 2013. Ten poster abstracts have been received of which eight made it into the final program. Each poster abstract has been reviewed by two members of the Performance 2013 student poster track's program committee.

Sincere thanks go to the members of the program committee who provided important reviews and helpful recommendations. Further cordial thanks go to Auguste Stastny for supporting the local organization of the poster session, and Ewald Hotop for quick Web updates whenever needed. Finally, many thanks go to the poster abstract submitters for contributing to the diverse program of the student poster session.

September 24, 2013

Karin Anna Hummel, ETH Zurich

Student Poster Program Chair

Student Poster Program Committee

Table of Contents

Cloudlet-based Mobile Offloading Systems: a Performance Analysis

Huaming Wu, Katinka Wolter
Department of Mathematics and Computer Science
Free University of Berlin, Berlin, Germany
{huaming.wu, katinka.wolter}@fu-berlin.de

Alessandro Grazioli
Information Engineering Department
Universit'a degli Studi di Parma, Parma, Italy
alessandro.grazioli81@gmail.com

***Abstract*—Offloading is an effective method to migrate programs from mobile devices to cloud, but it critically depends on network and cloud conditions. We suggest that the mobile device does not directly communicate with a distant cloud, but instead, with a nearby cloudlet acting as an intermediate node. Such a cloudlet-based offloading system is modeled and analyzed with respect to the state transition process with failures. In addition, we evaluated the failure and repair time, and four types of execution time as well. The numerical results reveal that in environments characterized by higher reachability of cloudlet and cloud, longer connection time or even larger speedup factor F, this scheme benefits from reduced execution time.**

***Index Terms*—offloading system; cloud computing; cloudlet; network bandwidth; performance analysis**

I. Introduction

Offloading a program from mobile devices to cloud is becoming increasingly attractive to reduce execution time and extend battery life. Apple's Siri and iCloud [1] are two remarkable examples. However, cloud offloading critically depends on a reliable end-to-end communication and on the availability of the cloud. In addition, it suffers from high network access latencies and low network bandwidth. Mahadev [2] proposed a vm-based cloudlet for the infrastructure setup of mobile systems. Instead, we want to investigate how effective and efficient they are and what factors influence their performance. With this purpose, we introduce a mathematical model and analyze cloudlet-based offloading systems with failures, considering application execution time and failure recovery time.

II. System overview

A. Problems concerning direct offloading systems

Network condition: Different network types have a large impact on communication time, cost and energy. 3G provides a near-ubiquitous coverage, but it consumes more energy than WiFi because of latencies, and is sensitive to location [3].

Cloud condition: Offloading is difficult in locations such as the interior of a tunnel or subway, where the low network bandwidth prevents cloud applications from working properly. In addition, distant cloud dependence could lead to severe problems when service outages occur.

B. Overview of cloudlet-based offloading systems

Rather than relying on a distant cloud, the resource poverty of a mobile device can be addressed by using a nearby resource-rich cloudlet via a wireless LAN. A cloudlet is a trusted, resource-rich computer which is well-connected to the internet and is available for use by nearby mobile devices [2]. As shown in Fig.1, cloudlets are dispersed and located close to mobile devices while cloud is generally far. At runtime, the app discovers a nearby cloudlet and offloads a computation-intensive program to it. The mobile device does not need to communicate with the distant cloud, but only with the cloudlet. This model decreases latency and lowers battery consumption by using WiFi instead of broadband wireless technology.

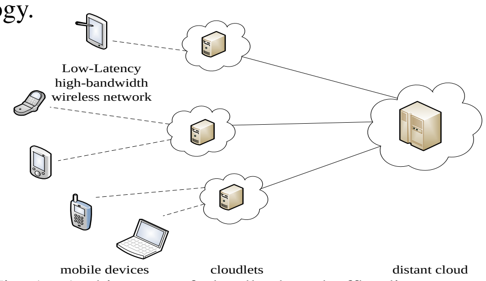


Fig. 1. Architecture of cloudlet-based offloading systems

Because wireless LAN bandwidth is remarkably higher than the bandwidth provided by radio access on a mobile device [3], we choose a path connecting the mobile device to a nearby cloudlet and then to a remote cloud. As depicted in Fig.2, D is the communicated data and B is the bandwidth between the mobile device and the cloud. Likewise, B_1 is the bandwidth between the mobile device and cloudlet, which generally uses gigabit internal connectivity and a high-bandwidth wireless LAN. B_2 is the bandwidth between the cloudlet and cloud, which is usually based on broadband technology. Generally, $B \leq B_1$ and $B \leq B_2$. Direct offloading saves execution time only if $T_\mathrm{m} > T_\mathrm{s} + \frac{D}{B}$, $T_\mathrm{m} = FT_\mathrm{s}$; T_m and T_s are the execution times on the mobile device and cloud, respectively; and the speedup $F \geq 1$ indicates how powerful the cloud is by comparison with the mobile device. Similarly, the cloudlet-based offloading saves time when $T_\mathrm{m} > T_\mathrm{s} + \frac{D}{B_1} + \frac{D}{B_2}$. Therefore, the cloudlet-based offloading model performs better than direct offloading approach when $\frac{1}{B} > \frac{1}{B_1} + \frac{1}{B_2}$.

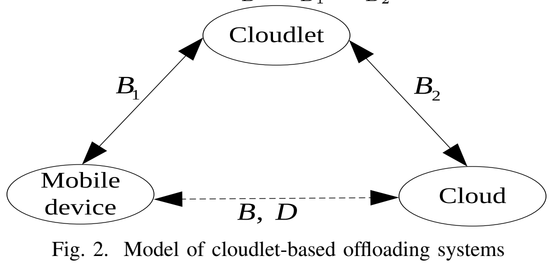


Fig. 2. Model of cloudlet-based offloading systems

III. Performance analysis

A. Ideal cloudlet-based offloading systems

For ideal offloading systems, no failure occurs. The pure program execution time is $T_\mathrm{p}(n) = (1-\sigma) \cdot T_\mathrm{m}(n) + \frac{\sigma \cdot T_\mathrm{m}(n)}{F}$ [5], here σ is the proportion between the sub-tasks performed by the cloud and the mobile device; $0 \leq \sigma \leq 1$, $(1-\sigma) \cdot T_\mathrm{m}(n)$ and $\frac{\sigma \cdot T_\mathrm{m}(n)}{F}$ represent the execution time spent on the mobile device and cloud, respectively; n is the number of sub-tasks. The total execution time is $T_\mathrm{OPT}(n) = T_\mathrm{p}(n) + T_\mathrm{c}(n)$, $T_\mathrm{c}(n) = \frac{D}{B_1} + \frac{D}{B_2}$ is the total communication time.

B. Cloudlet-based offloading systems with failures

There are four states depicted in Fig.3. When offloading starts, the execution state changes from S_{NE} to S_{OE1}. If the remote cloud is available, it changes to S_{OE2}. Once the executions of all offloaded components are successfully completed, the execution state changes back from S_{OE2} to S_{NE}. However, failures may occur in all states as [5]:

1) S_{NE}: running out of battery, abnormal shutdown.

2) S_{OE1}: wireless link failures, cloudlet shutdowns or becomes unreachable due to mobile device's movement.

3) S_{OE2}: cloud unavailable, cloud outages or becomes unreachable due to cloudlet's failures.

4) S_{FR}: nested failures may also happen.

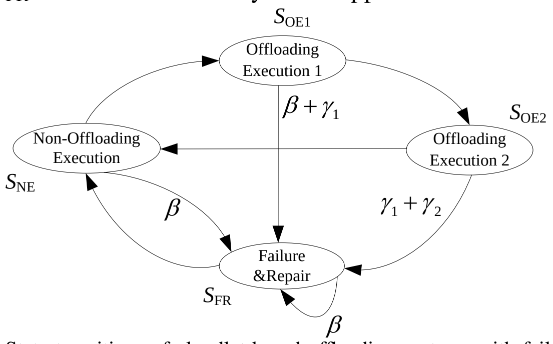


Fig. 3. State transitions of cloudlet-based offloading systems with failures

Offloading completes when the execution period elapses without failure. Independent failures caused by the mobile device, cloudlet and cloud are modeled as non-homogenous Poisson Processes with rates β, γ_1 and γ_2, respectively. According to Fig.3, the failure rate $\lambda(t)$ is defined as:

$$\lambda(t) = \begin{cases} \beta & \text{in state } S_{\text{NE}} \text{ and } S_{\text{FR}} \\ \beta+\gamma_1 & \text{in state } S_{\text{OE1}} \\ \gamma_1+\gamma_2 & \text{in state } S_{\text{OE2}} \end{cases} \tag{1}$$

The time period R is required to complete a repair in the presence of nested failures. The expectation of the failure repair time is $E(R^*) = \frac{1}{\beta}\left(\frac{1}{E[\mathrm{e}^{-\beta R}]} - 1\right)$ [5]. In presence of failures, the program execution time is calculated as

$$E[T_{\text{FT}}(n)] = \frac{1}{2}\Big[(1-\alpha_1)\cdot E[T_{\text{OE1}}(n)] + \alpha_1 E[T_{\text{NE/FR}}(n)] + (1-\alpha_2)\cdot E[T_{\text{OE2}}(n)] + \alpha_2 E[T_{\text{NE/FR}}(n)]\Big] \tag{2}$$

where $E[T_{\text{NE/FR}}(n)] = \left(\frac{1}{\beta} + E[R^*]\right)\left(\mathrm{e}^{\beta E[T_{\text{OPT}}(n)]} - 1\right)$ is the time spent in state S_{NE} and S_{FR}, $E[T_{\text{OE1}}(n)] = \left(\frac{1}{\beta+\gamma_1} + E[R^*]\right)\left[\mathrm{e}^{(\beta+\gamma_1)E[T_{\text{OPT}}(n)]} - 1\right]$ in state S_{OE1} and $E[T_{\text{OE2}}(n)] = \left(\frac{1}{\gamma_1+\gamma_2} + E[R^*]\right)\left[\mathrm{e}^{(\gamma_1+\gamma_2)E[T_{\text{OPT}}(n)]} - 1\right]$ in state S_{OE2}; α_1, α_2 are the probabilities of unreachability of cloudlet and cloud, respectively.

IV. Numerical results and analysis

If the program offloads, through a cloudlet, a number of sub-tasks to the cloud and the remaining ones are executed locally; we have execution time $T(n) = T_{\text{m}}[(1-\sigma)n] + T_{\text{FT}}(\sigma n)$, where $T_{\text{m}}[(1-\sigma)n]$ and $T_{\text{FT}}(\sigma n)$ are the times spent on the mobile device and cloud.

The parameters are set as follows: $E(R) = 10$, $n = 100$, $\sigma = 0.9$, $\beta = 10^{-3}$, $\gamma_1 = 10^{-4}$, $\gamma_2 = 10^{-5}$, $\alpha_1 = 0.1$ and $\alpha_2 = 0.2$. The average execution time for each sub-task on the mobile device is 5 second and $T_{\text{c}} = 0.3n$.

As shown in Fig.4, when F increases, the execution time decreases except for $T_{\text{m}}(n)$, which is horizontal at 500s.

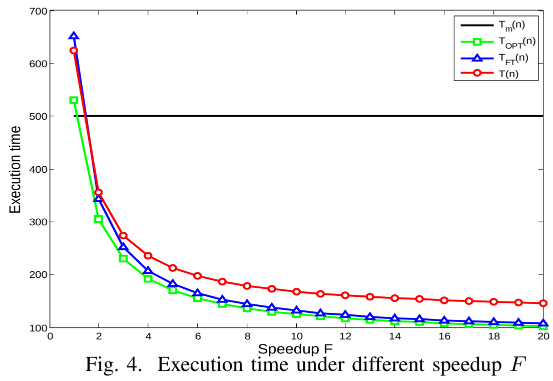


Fig. 4. Execution time under different speedup F

$T_{\text{FT}}(n)$ and $T(n)$ produce a remarkably longer time than $T_{\text{m}}(n)$ when $F < 2$, due to offloading and failure repairs. With larger F, the cloud can save execution time. However, time spent on offloading operation and failure handling will increase the total execution time, especially in higher cloudlet or cloud unreachability environments.

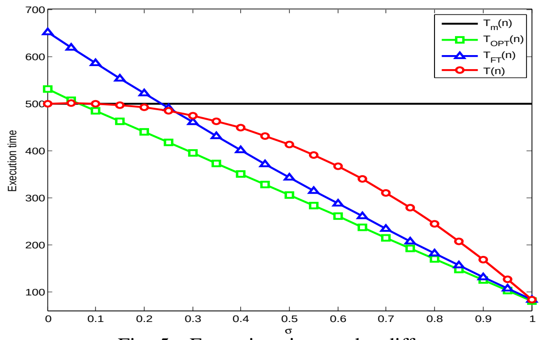


Fig. 5. Execution time under different σ

As shown in Fig.5, $F = 10$, $T(n)$ reduces to $T_{\text{m}}(n)$ when $\sigma = 0$ and $T(n)$ approaches $T_{\text{FT}}(n)$ when $\sigma = 1$. Both $T_{\text{FT}}(n)$ and $T_{\text{OPT}}(n)$ decrease along with the increase of σ, while $T(n)$ first decreases slightly and afterwards more rapidly with the increase of σ. $T_{\text{FT}}(n)$ is larger than $T_{\text{m}}(n)$ when $\sigma < 0.25$ due to offloading and failure recoveries. The larger σ is, the more execution time the cloudlet-based offload system saves.

V. Conclusion

In this short paper, we proposed an analytical model for cloudlet-based offloading systems, where the mobile device communicates with a nearby cloudlet instead of a remote cloud during the entire offloading process. In the environments characterized by high cloudlet or cloud unreachability, long disconnection time or even small speedup factor, this scheme will not benefit from reduced application execution time. The analysis results provide useful guidance for the design of efficient offloading systems.

References

[1] D. Niu, "Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications," *IEEE Wireless Communications*, vol. 3, 2013.

[2] M. Satyanarayanan, P. Bahl, R. Caceres and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14-23, October 2009.

[3] B. G. Chun and P. Maniatis, "Dynamically partitioning applications between weak devices and clouds," in *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing and Services: Social Networks and Beyond (MCS)*, no. 7, June 2010.

[4] K. Kumar and Y. H. Lu, "Cloud computing for mobile users: can offloading computation save energy?" *IEEE Computer*, vol. 43, no. 4, pp. 51-56, April 2010.

[5] S. Ou, Y. Wu, K. Yang and B. Zhou, "Performance analysis of fault-Tolerant offloading systems for pervasice services in mobile wireless environments," in *Proceedings of the 2008 IEEE International Conference on Communications (ICC'08)*, pp. 1856-1860, May 2008.

Comparative Analysis of Transfer Protocols For Big Data

[Extended Abstract]

Se-young Yu, Nevil Brownlee, and Aniket Mahanti
Department of Computer Science, University of Auckland, New Zealand
syu051@aucklanduni.ac.nz, {n.brownlee,a.mahanti}@auckland.ac.nz

ABSTRACT

Scientists working in various fields such as astronomy and biology require access to vast volumes of data (also referred to as "big data") for their research. This data may need to transferred over long distances to enable collaboration among scientists around the world. It is important for them to have appropriate transfer protocols to optimize data transfer speed. This paper presents a preliminary comparative analysis of high-speed transfer protocols for big data on long fat networks. We analyze the following protocols: GridFTP, FDT, UDT and Tsunami. We perform extensive experiments to measure the efficacy of each protocol in terms of its throughput for various round-trip times, against increasing levels of congestion inducing background traffic, and the impact of using jumbo frames sizes. Our results show that without much tuning, TCP-based protocols work better than UDP-based protocols.

1. INTRODUCTION

Data-intensive research stations such as the Large Hadron Collider, Square Kilometre Array, and International Thermonuclear Experimental Reactor produce huge amounts of data each year. This data may need to be transferred to distant locations using high-speed networks and analyzed by researchers globally. Most research stations use 10 Gb/s links to transfer their data, however, the transfer rates they achieve using standard protocols may still be relatively low.

To improve the performance of data transfer, providing only higher link capacity is not going to help much because the data transfer protocols have limitations. It is essential to choose an efficient data transfer protocol that can efficiently use the available capacity, and expedite the overall transfer process. Simply relying on TCP's congestion control may result in inefficient utilization of long fat pipes due to its conservative behaviour when handling packet loss events.

Understanding behaviour of TCP- and UDP-based high-speed data transfer protocols will allow researchers to choose a protocol that best suits their network environment, which may differ from one research site to another. Experimental evaluation of transfer protocols in a high-speed network is lacking in previous works.

In this paper, using a 10 Gb/s testbed network, we perform experiments with four well-known high-speed transfer protocols, namely, GridFTP,[1] FDT,[2] UDT,[3] and Tsunami.[4]. The former two protocols are TCP-based protocols and the later two protocols are UDP-based protocols. Our goal is to understand behaviour of each of these protocols over different Round-Trip Times (RTT), background traffic and the use of jumbo frames (i.e., frames of 9000 bytes instead of Ethernet's normal 1500 bytes). Furthermore, we want to compare TCP-based and UDP-based protocols, and see which protocols perform better under stated conditions.

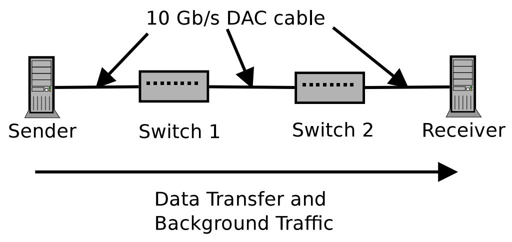


Figure 1: 10 Gb/s Testbed Network

Our preliminary analysis shows that TCP-based protocols are more efficient for large data transfers over a wide range of round trip times. When using jumbo frames, we notice increase in throughput of GridFTP on short-delay links.

2. METHODOLOGY

We built a testbed network in our lab with four machines (see Figure 1) to measure behaviour of a selection of high-speed data transfer protocols. Based on results from our initial experiments, we decided to measure performance of GridFTP with TCP, GridFTP with UDT, Tsunami and FDT. Since we did not have access to a hardware traffic generator, we chose to use an existing application program to generate (congestion-inducing) background traffic. We used `nuttcp` to generate background traffic. To observe their behaviour with various RTTs, we decided to use `netem`[5] to emulate different delays. We tuned the test machine's TCP parameters by following ESnet's Linux Tuning guide[6].

[1] `http://www.globus.org/toolkit/docs/latest-stable/gridftp/`
[2] `http://monalisa.cern.ch/FDT/`
[3] `http://udt.sourceforge.net/`
[4] `http://tsunami-udp.sourceforge.net/`
[5] `http://www.linuxfoundation.org/collaborate/workgroups/networking/netem`
[6] `http://fasterdata.es.net/host-tuning/linux/`

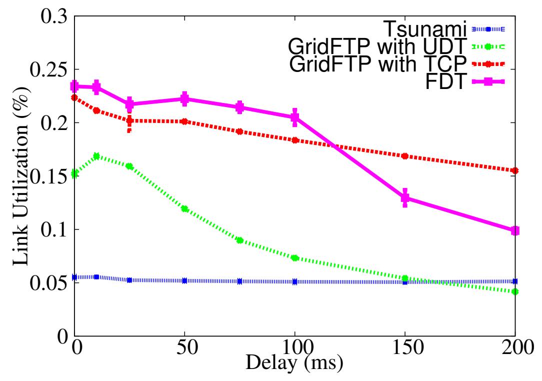


Figure 2: Throughput of the protocols using normal frames with varying RTT, with no background traffic [1]

3. RESULTS

We measure throughput of each protocol when they are used to transfer data in our testbed network with varying RTT as well as varying level of background traffic with varying RTT. We also measure the impact of using jumbo frames.

3.1 Impact of increasing distance

Figure 2 shows how throughput varies with RTT for the four protocols. We observe that FDT performs very well for short RTT links, while GridFTP performs better with relatively longer RTT links.

Without congestion, performance of FDT and GridFTP with TCP is higher than Tsunami and UDT. The highest throughput of FDT is 2.34 Gb/s with a 1 ms RTT but it decreases rapidly after 100 ms compared to GridFTP, which performs better than FDT when link RTT is longer than 100 ms. Interestingly, the throughput of Tsunami did not decrease over increasing RTT, showing it has the most effective congestion control with increasing RTT.

3.2 Impact of jumbo frames, background traffic, and increasing RTT

Figure 3 shows how throughput varies when using jumbo frames with increasing RTT for GridFTP with TCP, GridFTP with UDT and FDT.

With increasing UDP background traffic, FDT performs very well, while GridFTP with TCP and GridFTP with UDT perform poorly. Tsunami did not work with jumbo frames and background traffic, so we did not consider it further. With jumbo frames, the rate of decrease in throughput over increasing delay for FDT is reduced. As a result, FDT performs as well over long RTT links as it did for those with short RTT links. Increasing background traffic did not cause much problem for FDT and GridFTP with UDT, but it made GridFTP with TCP unstable. Using jumbo frames increased the throughput of GridFTP for short RTT links, but as RTT increases, throughput of GridFTP did not improve much compared to the results with normal-size frames.

4. CONCLUSIONS AND FUTURE WORK

We presented a preliminary comparative analysis of four well-known transfer protocols for big data, namely GridFTP, FDT, UDT, and Tsunami. Our results show that using GridFTP with TCP or FDT gives higher throughput over a 10 Gb/s network link. Even with significant level of congestion induced by background traffic, FDT was stable enough to perform data transfer. On the other hand, UDP-based protocols did not perform well in most tests and we do not recommend using Tsunami due to its instability when using jumbo frames or when faced with background traffic. Our results can be used by scientists to choose the most appropriate protocol for transferring their data over long distances.

There are several avenues for future work. During the experiment, we have found difficulties with generating enough TCP background traffic with generally accepted traffic generator including `Iperf`, `Netperf`, and `nuttcp`. When the number of TCP flows increases, the flows interfere with each other. There are other aspects of protocols to be measured such as TCP friendliness or effect of using multiple concurrent connections. We expect to perform a further measurement for such aspects in the near future We also plan to expand our testbed over long haul links, so as to determine how best to transfer big data over long distances over production networks.

5. REFERENCES

[1] S. Yu, N. Brownlee, and A. Mahanti. Comparative performance analysis of high-speed transfer protocols for big data, Apr. 2013. submitted for publication.

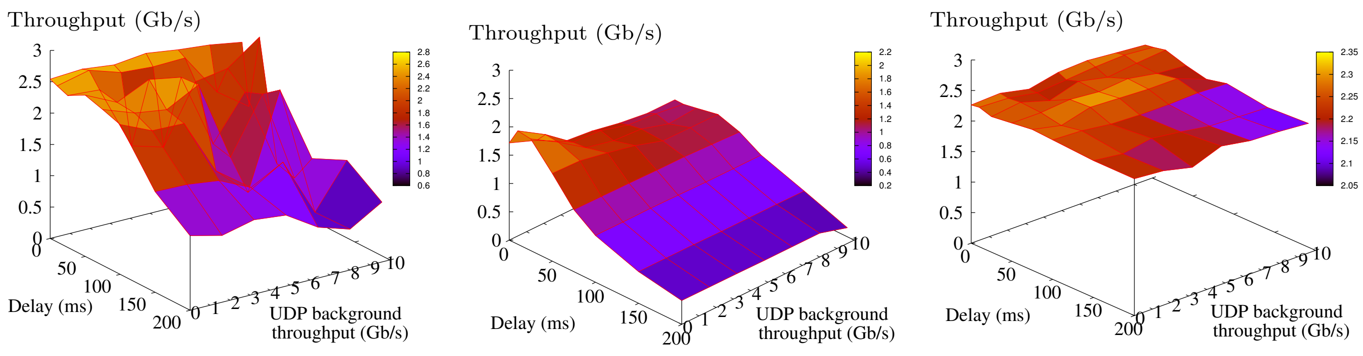


Figure 3: Throughput of GridFTP with TCP (left), GridFTP with UDP (middle), and FDT (right) using jumbo frames with increasing RTT and UDP background traffic

Dynamic Resource Provisioning for Concurrent MapReduce Frameworks

Bogdan Ghiţ
Delft University of Technology
the Netherlands
b.i.ghit@tudelft.nl

Dick Epema
Delft University of Technology
the Netherlands
d.h.j.epema@tudelft.nl

ABSTRACT

Running multiple instantiations of the MapReduce framework (MR-clusters) concurrently in a multicluster system or data center enables workload and data isolation, which is attractive for many organizations. We provision MR-clusters such that they receive equal levels of service by assigning each such cluster a dynamically changing weight that indicates its fair share of the resources.

1. INTRODUCTION

Despite the high scalability of the MapReduce framework in large infrastructures such as multicluster systems and data centers, *isolating* MapReduce workloads and their data is very attractive for many organizations. In this paper, we propose a *dynamic* resource management approach for provisioning multiple MR-clusters in a single multicluster or data center, such that distinct MR-clusters receive equal levels of service. Towards this end, we differentiate each MR-cluster by assigning a dynamically changing weight to it that indicates its fair share relative to all active MR-clusters.

Running multiple MR-clusters concurrently within the same physical infrastructure enables *four types of isolation*, viz. with respect to performance, to data management, to fault tolerance, and to version [1]. Deploying each MR-cluster on a static partition of the system may lead to poor resource utilization, as some MR-clusters may be accessed more frequently than others, creating an imbalance between the levels of service they receive. To dynamically change the resource allocations at runtime, we need to understand which factors can be used to differentiate the MR-clusters. Assuming no prior knowledge about the workloads, we propose three factors that can be used to establish the fair shares of the MR-clusters: the size of their workloads, the utilization of their allocated resources, and their service levels.

As MapReduce is usually employed for data-intensive applications, one of the main issues is to limit the overhead of reorganizing the data when *growing* or *shrinking* an MR-cluster when its share changes.

2. BACKGROUND

In this paper we take an experimental approach to resource provisioning MR-clusters. The testbed for our experiments is the multicluster system DAS-4 [1], which is a wide-area computer system dedicated to research in parallel and distributed computing that is currently in its fourth generation and that consists of six clusters distributed in institutes and organizations across the Netherlands.

KOALA [3] is a grid resource manager developed for multicluster systems such as the DAS-4 with the goal of designing, implementing, and analyzing scheduling strategies for various application types (e.g., map-reduce, cycle scavenging jobs, workflows). We have recently extended KOALA with a MapReduce runner [1], which is a specialized module for scheduling and deploying MR-clusters on demand.

[1]http://cs.vu.nl/das4

3. FAIR RESOURCE ALLOCATIONS

Our basic goal when provisioning resources to multiple MR-clusters in a single multicluster or data center is to give them equal levels of service (e.g., throughput, response time). In order to achieve this, we want to assign each MR-cluster a dynamically changing weight that indicates the share of the resources it is entitled to.

3.1 System Model

When growing or shrinking an MR-cluster, both the storage and compute layers need to be adjusted. Consequently, we distinguish three types of nodes in the system. The *core nodes* are fully functional nodes allocated for the MR-cluster deployment, with components in both layers of the MR-cluster. The *transient nodes* are temporary compute nodes used to grow MR-clusters after their initial deployment. Removing transient nodes does not change the distribution of the data. The *transient-core nodes* are temporary fully functional nodes, also used to grow active MR-clusters. Their removal requires replicating the data they locally store.

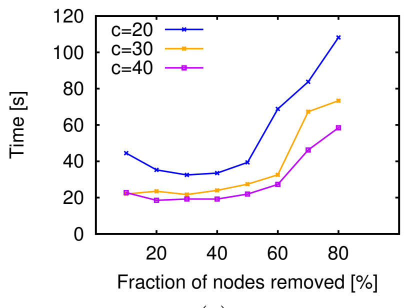
3.2 Notion of Fairness

The MR-clusters are entitled to shares of the total data center capacity proportional to their given weights. For an MR-cluster i, the difference between the fraction $r_i(t)$ of resources it currently has and the share of resources it should have based on its weight $w_i(t)$ at time t is defined as its *temporal discrimination* [2]: $D_i(t) = r_i(t) - w_i(t)$.

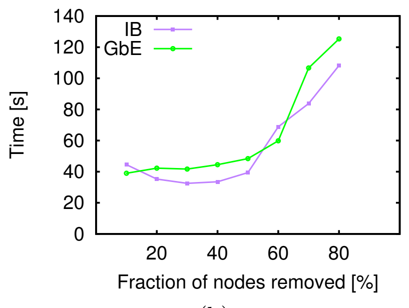
We define the discrimination of MR-cluster i during a time interval $[t_1, t_2]$ by

$$D_i(t_1, t_2) = \int_{t_1}^{t_2} (r_i(t) - w_i(t))dt. \qquad (1)$$

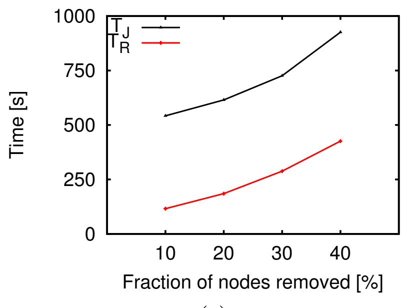
Setting $t_1 = a_i$ and $t_2 = d_i$ with a_i and d_i the moments of the request for the deployment and the shutdown of the MR-cluster, respectively, we obtain the *overall discrimination* of the MR-cluster. The fairness of the system over a certain interval is indicated by the *global discrimination*, which is



(a)



(b)



(c)

Figure 1: The average per-node reconfiguration time depending on the fraction of removed nodes for (a) MR-clusters with c core nodes and for (b) two networks on MR-clusters with 20 core nodes. The job execution time T_J and the reconfiguration time T_R when removing different fractions of nodes from a 20-node MR-cluster running a WordCount job, after 25% of the map tasks have completed (c).

defined as the variance $Var(D)$ across all active MR-clusters during that interval.

3.3 Policies for Setting Weights

We define three sets of policies for setting the weights of active MR-clusters, which are based on: the demands of the workloads submitted to them (*demand-driven*, e.g., in terms of the numbers of jobs or tasks in queue), the usage of the resources they are currently allocated (*usage-driven*), and the service they obtain in terms of one of our metrics (*service-driven*). In all of these policies, the weights of the MR-clusters and the global discrimination are recomputed for every interval of length T. Only when the global discrimination exceeds a threshold τ are the allocations of the MR-clusters actually changed according to the new weights.

3.4 Resizing Active MR-clusters

MR-clusters with positive discrimination have to shrink their resources. There are two options for doing so: non-preemptive, with the data replication running in parallel with the workload execution, or preemptive, with the shrinking phase starting after the running tasks are killed. MR-clusters with negative discrimination have to grow their resources. Based on the type of additional nodes used, there are two options: growing with transient or with transient-core nodes.

4. EXPERIMENTS

In our experiments, we assess the performance of our three sets of policies for different values of T and τ with (non-)preemptive shrinking and core/transient growing. As a basis, we perform a set of *micro-experiments* in which we assess for instance the reconfiguration overhead when shrinking an active MR-cluster, and the performance of running single MapReduce applications based on the number of core and transient nodes allocated to it. For the former micro-experiment, we set up MR-clusters of different sizes with 10 GB per node replicated 3 times. We find that the cluster size and the fraction of nodes removed have a significant impact on the reconfiguration time (Figure 1a). Furthermore, increasing the network bandwidth to 20 Gb/s improves the reconfiguration time with only less than 20% (Figure 1b). Shrinking an active MR-cluster while running a job increases the reconfiguration time (Figure 1c). For the latter micro-experiment, we set up MR-clusters of different fractions of core nodes and we run two common MapReduce applications. We find that WordCount scales better on transient nodes than Sort (Figure 2).

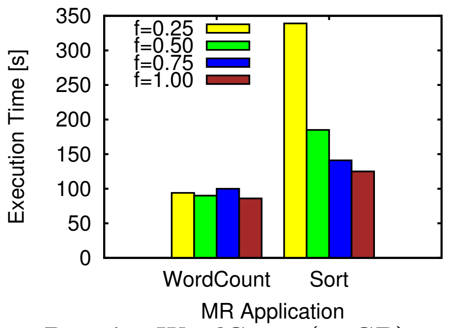


Figure 2: Running WordCount (40 GB) and Sort (50 GB) on 40-node MR-clusters with different fractions f of core nodes.

5. CONCLUSION

Dynamic resource provisioning to multiple instantiations of the MapReduce framework deployed in single multiclusters or data centers is of both practical and theoretical interest. In this paper we propose a form of such provisioning that targets equal levels of service for the active MR-clusters.

6. ACKNOWLEDGMENT

This work was partially funded by the Dutch national research program COMMIT.

7. REFERENCES

[1] B. Ghit, N. Yigitbasi, and D. Epema. Resource Management for Dynamic MapReduce Clusters in Multicluster Systems. In *IEEE High Performance Computing, Networking, Storage and Analysis (SCC), SC Companion, 2012.*

[2] D. Raz, H. Levy, and B. Avi-Itzhak. A Resource-Allocation Queueing Fairness Measure. In *SIGMETRICS 2004.*

[3] O. Sonmez, H. Mohamed, and D. Epema. On the Benefit of Processor Co-Allocation in Multicluster Grid Systems. *IEEE Trans. on Parallel and Distributed Systems*, 21:778–789, 2010.

Intelligent Decision-Making in Distributed Dynamic VM Consolidation Using Fuzzy Q-Learning

Seyed Saeid Masoumzadeh
Research Group Entertainment Computing
University of Vienna
Vienna,Austria
masoumzadeh@gmail.com

Helmut Hlavacs
Research Group Entertainment Computing
University of Vienna
Vienna,Austria
helmut.hlavacs@univie.ac.at

ABSTRACT

A cloud manager deals with a dynamic multi objective optimization problem. Indeed, this problem lies in the fact that there is always a tradeoff between energy and performance in a virtualized data center. Therefore, a cloud manager must be equipped with a strategy to consolidate virtual machines and configure them dynamically in a way that optimizes energy-performance tradeoff in an online manner. Distributed dynamic VM consolidation strategy can be an effective one to tackle this problem. The procedure of this strategy can be decomposed into four decision-making tasks.1) Host overloading detection; 2) VM selection; 3) Host underloading detection; and 4) VM placement. The dynamic optimization is achieved when each of aforementioned decisions are made optimally in an online manner. In this paper with concentration on host overloading detection and VM selection task, we propose the Fuzzy Q-Learning (FQL) as an intelligent and online machine learning approach in order to make optimal decisions towards dynamic energy-performance tradeoff optimization.

Categories and Subject Descriptors

D.4.7 [**Operating Systems**]: Organization and Design—*Distributed systems*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligence Agent,Multiagent systems*

General Terms

Algorithms,Management

Keywords

Dynamic VM Consolidation, Energy Efficient Cloud Manager, Fuzzy Q-Learning

1. INTRODUCTION

Unfortunately, server consolidation enabled by virtualization introduces a new problem to the cloud environment. Since the size of Virtual Machines (VMs) inside a physical server might change due to dynamic workloads, resource underutilization or overutilization might occur, causing either inefficient energy consumption or unwanted performance degradation. Consequently, the cloud managing system needs to employ a strategy to reallocate resources to virtual machines by live migration dynamically in order to maximized per-server utilization while attaining the promised non-functional qualities of the service guaranteed in the Service Level Agreements (SLA). In other word, this strategy must be able to optimize energy-performance trade-off in an online manner. Distributed dynamic VM consolidation can be an effective strategy to tackle this problem [1]. The procedure of this strategy can be decomposed into four decision-making tasks: (1) Host overloading detection, here deciding when a host must be considered as overloaded. In this situation, one or more VMs must be migrated away from this host. (2) Host underloading detection, here deciding when a host must be considered to be underloaded. In this situation, the host is ready to switch to sleep mode and all VMs must be migrated away from this host. (3) VM selection, here deciding, which VMs should be migrated away from overloaded hosts, and (4) VM placement, here deciding about which host must be selected to receive migrating VMs. Indeed, the energy-performance trade-off optimization is achieved when each of aforementioned decisions are made optimally in an online manner.

Q-learning as a model free Reinforcement Learning (RL) is an effective approach to design dynamic system managements and consequently produce dynamic decision-making tasks. In principle, Q-learning can automatically learn high-quality decision making tasks without an explicit model, and with little or no built-in system specific knowledge. Fuzzy Q-Learning (FQL) [2] is the fuzzy extension of Q-Learning. It is capable of handling continuity in the state space; in addition, it is powerful enough to tackle the curse of dimensionality and other ordinary RL issue rising in real life and industrial problem. In this work we employ FQL approach in order to give an intelligent behavior to the physical hosts for making optimal decision into the dynamic VM consolidation procedure towards energy-performance tradeoff optimization. Several researches have been done so as to enhance the dynamic VM consolidation and several algorithms and policies have been proposed for host overloading detection and selecting VM until now. In contrast to previous studies, our proposed approach can achieve better result in improving energy-performance tradeoff in cloud data center.

2. DATA CENTER AND FQL AGENTS

In our data center each physical host is associated with two FQL agents. One of them is responsible for making decision about when a host must be considered as an overloaded host and another one is responsible for making deci-

IFIP WG 7.3 Performance 2013, September 24-26, Vienna, Austria. Copyright is held by author/owner(s).

sion about which VM must be selected to migrate when a host is overloaded. All the agents inside data center by interacting whit their own corresponding host learn how to make decisions and consequently take actions according to dynamic characteristics of physical host state towards energy-performance trade-off optimization. Since the agents inside data center may experiment similar states to make decision during their lifetime; thus, we propose a cooperative learning strategy by sharing learning knowledge among agents. It can help to increase learning convergence rate and consequently make better decisions.

3. INTELLIGENT DECISION-MAKING IN HOST OVERLOADING DETECTION

Each learning agent i at each time-step t, observes the current state of its own corresponding host S_{t_i}, and gathers some information including average CPU utilization and number of VMs residing on the host as the input state X_{t_i}, where $1 \leq i \leq N$ and N is the number of hosts inside the data center. Further, the learning agent immediately chooses a threshold value as an action from the actions set $A(X_t)$. The threshold value is used in the host overloading detection procedure, if the CPU utilization exceeds the threshold; it means the host is overloaded and one or more VMs must be selected to migrate. One time-step later, in parts of a consequence action, the FQL agent receives a numerical reward, $Reward_{i_{t+1}}$ and finds itself in a new state S_{t+1_i}. The reward is a feedback on the resulted new threshold value from the actions set. The feedback is provided by a tradeoff model (we will address it in section 5) that is able to capture both energy and performance at each interval in a host. In a trial and error interaction, finally the FQL agent learns how to map the sates to the actions, so as to maximize the discounted sum of reward obtained. In other words, the FQL agent learns the best sequence of selecting thresholds, corresponding to the states observed during its lifetime.

4. INTELLIGENT DECISION-MAKING IN VM SELECTION

When a host becomes overloaded, a VM selection task is called so as to remedy this situation by choosing one or more VMs for migration. Several criteria have been proposed to select VMs in the previous literatures until now .The FQL agent is able to make decision about which criterion can be effective at each state of its corresponding host for selecting VM towards energy-performance tradeoff optimization. The FQL procedure is like what we discussed in the previous section; however, the definition of action set is different. In fact, each element of the actions set $A(X_t)$ is a criterion for selecting VM, such as minimum CPU utilization and maximum CPU utilization. The FQL agent in a trial and error interaction learns the best sequence of criteria, corresponding to the states observed during its lifetime.

5. ENERGY-PERFORMANCE TRADE-OFF MODEL

To represent the energy-performance trade-off, we use the product combination of SLA Violation (SLAV) and energy consumption (EC). The SLAV [1] is a metric , which can

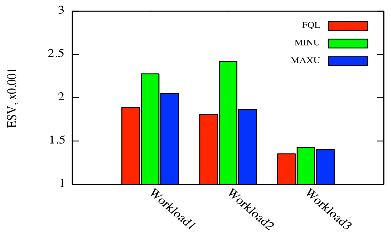


Figure 1: Comparision of VM slection strategies

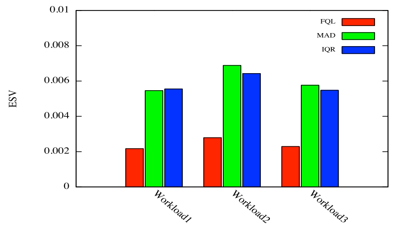


Figure 2: Comparison of host overloading detection strategies

encompass both performance degradation due to host overloading and VM migration in each host.The EC [1] is defined as a linear function of the CPU utilization of each host.

$$ESV_i = SLAV_i \times EC_i \qquad 1 \leq i \leq N \tag{1}$$

6. EXPERIMENTAL RESULTS

We used CloudSim toolkit to evaluate our experiments. At the first experiment, we have compared the result of the dynamic VM consolidation procedure when its VM selection task uses the FQL strategy with ones use minimum utilization and maximum utilization as a fixed criterion. At the second experiment, we have compared the result of the dynamic VM consolidation procedure when its host overloading detection task uses the FQL strategy with the state of the art algorithms [1] that use some adaptive heuristics, based on statistical analysis of historical data for estimating CPU utilization threshold. In the comparison we have measured the ESV (Energy SLA Violation) value [1]. These experiments have been evaluated by three different workloads in tree different days. The results of these experiments have been illustrated in figure 1 and 2 respectively.Both results show the superiority of the FQL as an intelligent decision making strategy in improving energy-performance trade-off in comparison with the state of the art strategies.

7. REFERENCES

[1] A. Beloglazov and R. Buyya, “Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers,” *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[2] L. Jouffe, “Fuzzy inference system learning by reinforcement methods,” *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 28, no. 3, pp. 338–355, 1998.

Energy-efficient Execution of Mobile Services using Flexible Cloud Platforms

Rene Gabner*
gabner@ftw.at

Hans-Peter Schwefel*†
schwefel@ftw.at, hps@es.aau.dk

*Telecommunication Research Center Vienna
Donau-City-Straße 1
1220 Vienna, Austria

†Aalborg University / Department of Electronic Systems
Niels Jernes Vej 12
9220 Aalborg, Denmark

ABSTRACT

Two recent trends are major motivators for service component migration: the upcoming use of cloud-based services and the increasing number of mobile users accessing the Internet-based services via wireless networks. While cloud-based services target the vision of Software as a Service, where services are ubiquitously available, mobile use of such services impairs the battery life of the device caused by additional network communication between the cloud and the mobile device. This paper investigates service component migration between the mobile client - as part of the cloud - and the infrastructure-based cloud as a means to save energy of the mobile device. Hereby, migration decisions are controlled by policies. A Markov modelling approach to calculate optimised policies with respect to energy consumption at the client is introduced and applied to an example scenario.

1. INTRODUCTION

Mobile cloud usage is challenged by varying wireless link conditions, as well as by limited resources at the mobile device. Therefore energy consumption of a device connected to the cloud is an important issue, both execution of a service at the client, and network communication from/to the infrastructure-based cloud have to be considered.

An approach to dynamically migrate service components between the mobile device and the infrastructure-based cloud can alleviate these energy consumption problems. In doing so, the trade-off between increased energy consumption while executing components at the client, energy costs required for the component migration and reduced communication needs during execution has to be carefully balanced.

While there are several frameworks e.g. [1] that allow to flexibly execute applications between mobile devices and a cloud environment, the evaluation of such frameworks in simulation models [7], experimental prototypes [4], and analytic models [5] is based on heuristic choices of component migration policies. Wen et al. [6] determine optimised choices restricted to an individual application component in the setting of CPU clock frequency optimisation at the mobile device and remote invocation costs only. In contrast to that we consider real-time relocation of the whole set of components that constitute the application. The general description of a model for computation of optimised policies with respect to application execution time has been presented in previous work of the authors [2][3]. This paper extends the modelling approach to enable a new target metric for optimisation: the probability to finish within a given energy budget.

2. SYSTEM DESCRIPTION AND EVALUATION MODEL

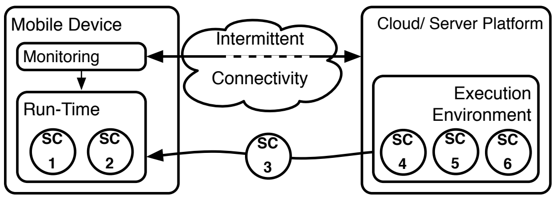


Figure 1: Overall system architecture.

The primary target of the dynamic component placement is to decrease the energy consumption at the mobile device. Therefore we adopt the approach presented in [3] to model energy consumption instead of execution time. The application is decomposed into service components, which timings are modelled using a continuous time Markov chain. All components are executed in a sequential order. We introduce a monitoring component at the client, which observes the service progress of the components, as well as the energy consumption of the device. The reconfiguration of the system (shift of service components from or to the client device) is triggered by the energy level observed by the monitoring component at the client and based on a pre-defined policy defining the target placement of all relevant application components. Such a reconfiguration process is depicted in Figure 1, where service component SC3 is just being moved from the cloud to the mobile device.

We assume a factor k_{client} to model the slower execution of components at the client; a mean delay D_{rem} to model additional delay for remote component calls; D_{fail} for delay in case of link failures (i.e. the time to recover the communication link), p_f for the probability of the network to fail, and *budget* to set a maximum energy budget. The effort to transfer service components between client and cloud is modelled by a *cost* value which introduces additional delay and thus a certain energy budget, which could depend on the component size.

Figure 2 shows the model of remote calls. States i and j represent local and remote execution of components. The transition from state i to state j with rate q_{ij} in the application model is in this scenario replaced by the transition structure via the 2 states in the lower part of Figure 2. Those 2 states $(ij)_{up}$ and $(ij)_{down}$ model the network states for remote calls where execution is passed to the remote component and the case of broken communication links.

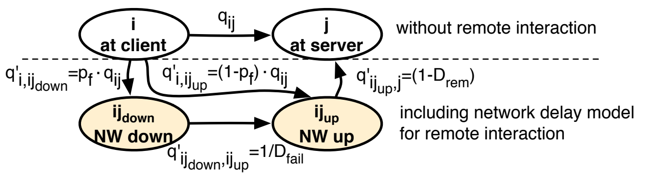

i
at client
q_{ij}
j
at server
without remote interaction
$q'_{i,ij_{down}}=p_f \cdot q_{ij}$
$q'_{i,ij_{up}}=(1-p_f)\cdot q_{ij}$
$q'_{ij_{up},j}=(1-D_{rem})$
ij_{down}
NW down
ij_{up}
NW up
including network delay model for remote interaction
$q'_{ij_{down},ij_{up}}=1/D_{fail}$

Figure 2: Extended service component model including network failure and execution time [2].

3. APPLICATION TO ENERGY CONSUMPTION METRICS

The four different states in Figure 2 correspond to different types of activities at the mobile device and hence are associated with different power levels:

- **IDLE** (100mW): In state j in the example, no service computation and no messaging takes place at the client. The mobile device can operate in an energy-efficient low power mode only running the necessary background processes and hence has a low power consumption.
- **COMPUTE** (200mW): In this state, service component execution (computing) is carried out by the mobile device, while no communication takes place. Note that depending on the actually used resources on the mobile device, this single state, i in the example, could be replaced by a network of states. For the numerical example later, we however assume a single power value.
- **TRANSCEIVE** (600mW): In this state, the device is assumed to be transmitting or receiving, i.e., either due to migration of components or due to remote service component invocation. The example $(ij)_{up}$ in Figure 2 would actually be a transmitting state (as the control flow passes from a client component to a server component). For the numerical examples later, we use the same power level for transmitting and receiving.
- **TRYING** (800mW): In this state, $(ij)_{down}$ in the example, the mobile device tries to transmit, but due to some link or network problems, it is not possible. This state includes both searching for a channel, a busy physical layer, and potential retransmission attempts.

To transform the time model to the energy model, we scale the transitions out of the 4 states in Figure 2 according to power values for each state. The former time instant of reconfiguration, called γ in [3], now translates to 'consumed energy at the mobile device' in the scaled model. The deadline on application execution time in [3], now becomes a total available energy budget for this considered multi-component application at the device. The mathematics to calculate the probability of finalising the application within the energy budget under a potential reconfiguration at energy consumption γ remains analog to the time-based model and also the computationally efficient search for an optimised target configuration remains analogously.

4. NUMERICAL RESULTS

We present a result graph showing the probability for application execution within the energy budget in Figure 3. The service consists of six components with a mean execution time of one second per component, where the initial configuration is: SC1-3 at the client and SC 4-6 at a server. The x-axis is the reconfiguration moment, measured by consumed energy so far at the device. The solid curve shows a static case without reconfiguration while the dashed curve represents a reconfiguration to an optimal target configuration that was determined by the Markov model evaluation itself. So both the policy optimisation and the performance calculation are done via the model introduced in the previous section. The parameters of the scenario are shown in the title of Figure 3 and have been explained in Section 2 and 3 respectively.

After the service used 3.2 J, it is the optimal time to reconfigure the service, which leads to an increase of the probability to finish the service within the given energy budget of 30 J by 12%.

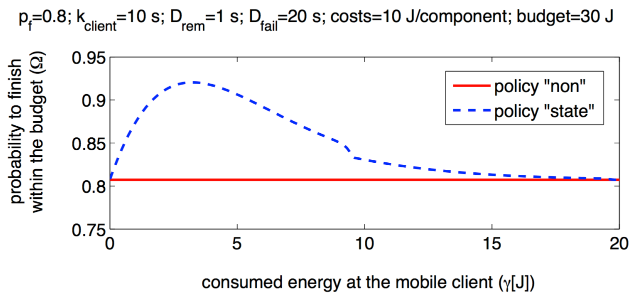


Figure 3: Increased probability to finish the service in case of delayed reconfiguration

5. REFERENCES

[1] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. Maui: making smartphones last longer with code offload. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 49–62, New York, NY, USA, 2010. ACM.

[2] R. Gabner, K. A. Hummel, and H.-P. Schwefel. Modeling movable components for disruption tolerant mobile service execution. *Diaz, M.; Avresky, D.; Bode, A.; Bruno, C.; Dekel, E. (Eds.) CloudComp 2009. LNCS*, 34(1):231–244, 2010.

[3] R. Gabner, H.-P. Schwefel, K. A. Hummel, and G. Haring. Optimal model-based policies for component migration of mobile cloud services. In *NCA*, pages 195–202, 2011.

[4] T.-Y. Lin, T.-A. Lin, C.-H. Hsu, and C.-T. King. Context-aware decision engine for mobile cloud offloading. In *Wireless Communications and Networking Conference Workshops (WCNCW), 2013 IEEE*, pages 111–116, 2013.

[5] S. Ou, Y. Wu, K. Yang, and B. Zhou. Performance analysis of fault-tolerant offloading systems for pervasive services in mobile wireless environments. In *Communications, 2008. ICC '08. IEEE International Conference on*, pages 1856–1860, 2008.

[6] Y. Wen, W. Zhang, and H. Luo. Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones. In *INFOCOM, 2012 Proceedings IEEE*, pages 2716–2720, 2012.

[7] S. Yang. Manageable granularity in mobile application code offloading for energy savings. In *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*, pages 611–614, 2012.

Performance of Epidemic Content Distribution in Opportunistic Networks

Ljubica Pajevic, Gunnar Karlsson, Ólafur Helgason
KTH Royal Institute of Technology
Linnaeus ACCESS Center
Stockholm, Sweden
{ljubica, gk, olafurr}@kth.se

ABSTRACT

Epidemic spreading is a common approach to mitigate frequent link disruptions and to support content-centric information dissemination in opportunistic networks. Stochastic models, often used to characterize epidemic processes, introduce assumptions which, on one hand, make them analytically tractable, while on the other, ignore attested characteristics of human mobility. We investigate the fitness of a simple stochastic model for content dissemination by comparison with experimental results obtained from real-life mobility traces. We examine four mobility datasets and consider content delivery delay as a performance metric. Our finding is that a homogeneous model is unable to capture the performance of content dissemination with respect to content delivery delays.

Keywords

Epidemic modeling, content distribution, opportunistic networks, ad hoc networks

1. INTRODUCTION

In opportunistic networks, highly dynamic network topology and intermittent connectivity make routing a challenge. A large body of algorithms for opportunistic networking employ epidemic spreading principle. Adopting the principles of epidemic modeling from the field of mathematical biology, stochastic modeling has become a method commonly used in networking. However, for the sake of analytical tractability, models often assume identical mobility and contact patterns for all nodes in the network. Recently, studies which consider heterogeneous networks have been emerging; yet, it is debatable whether there are savings in complexity and increased understanding from using the models compared to simulations.

In this poster, we present an empirical study of epidemic content spreading by using real-world mobility traces. Then, we consider an analytic model proposed in [3], and investigate if a homogeneous model can be utilized to evaluate the performance of opportunistic networks when the assumptions on network homogeneity are relaxed.

2. OPPORTUNISTIC CONTENT DISTRIBUTION: MODEL AND EVALUATION

The application scenario we consider is that of disseminating information by utilizing opportunistic contacts, based on user interest. Sharing local news, traffic and tourist information in public areas, public announcements at massive events, or mobile advertisements are common examples where this distribution scheme can be used. We are interested in evaluating the performance of spreading in terms of delivery delays.

Epidemic model

We consider a homogeneous set of N mobile nodes, moving in a confined area and exchanging information through intermittent contacts. Initially, a single node carries a piece of information (a content item) and $N-1$ nodes are interested in obtaining the contents. We assume that the mobility of nodes is such that inter-contact times between any two nodes can be modelled by identical, independent and exponentially distributed random variables with rate λ. This allows analysis of the epidemic process with a stochastic model based on a continuous-time Markov chain, described in [3].

To characterize the performance, we look at two metrics: the time until the information has reached all the N nodes in the network, denoted by *overall delivery time* T_{odt}, and the *individual delivery time* T_{idt} , defined as the time until an arbitrary node has obtained the contents. Their expected values, $E[T_{odt}]$ and $E[T_{idt}]$ can be found from expressions given in Tab. 1.

Table 1: Expected delivery times

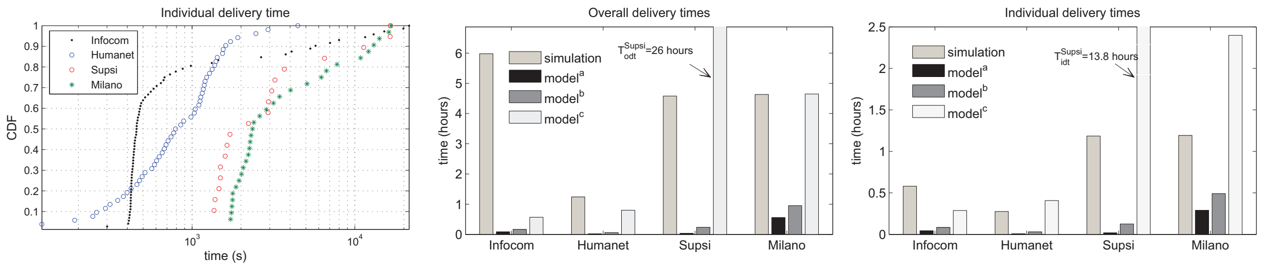
Overall delivery time $E[T_{odt}]$	$\frac{2}{N\lambda}H_{N-1}$
Individual delivery time $E[T_{idt}]$	$\frac{1}{\lambda(N-1)}H_{N-1}$

$H_n = \sum_{i=1}^{n} 1/i$ is the n-th harmonic number

Mobility Traces

To cover various scenarios, we use four experimental datasets, diverse in time granularity, number of participants in the experiments and in the experiment duration. The datasets report direct pair-wise contacts between users moving in relatively restricted areas: a conference venue, a university, or a company building. We briefly describe the contexts where the traces were collected and the acquisition methods used, and our methodology of pre-processing the traces.

- **Infocom** traces [5], obtained at Infocom 2006, report contacts between 78 experiment participants carrying iMote devices during four days. The scanning interval was 120 seconds.
- **Humanet** traces [1] describe human mobility in an office scenario, reporting proximity traces of 52 participants during one working day. The users were carrying Bluetooth customized devices, which were scanning every 5 seconds to capture direct contacts with other devices.
- **Supsi** dataset [2] comprises of contacts in a group of 39 participants, from three institutes, located in two buildings. The participants were carrying sensor nodes with



(a) (b) (c)

Figure 1: CDFs of the contents delivery times (a). Comparison of the overall (b) and individual delivery times (c).

a transmission range of 5 meters and a scanning cycle of 10 milliseconds.

- **Milano** dataset [4] was obtained by 44 participants. Contacts were logged by customized radio devices, operating with a transmission range of 10 meters and a configurable scanning interval of one second.

The duration of the experiments varies from a single day (*Humanet*) to several weeks (*Supsi*); to obtain inter-contact times, we consider only the days when a considerable number of contacts were recorded, and observe only the contact events that occurred during work hours (usually from 9:00 to 18:00).

Evaluation

We want to assess the capability of a homogeneous epidemic model to capture the process of epidemic content spreading in real-life scenarios. First, we simulate the epidemic process in four scenarios, and plot the CDFs of the contents delivery times in Fig. 1(a). For each of the traces, we choose a single day when the nodes were most active, seen as the number of contacts recorded during that day. The analytic model can be seen as a simple and efficient tool to estimate the performance; its simplicity stems from the fact that it requires only two input parameters: the number of nodes in the network and the node contact rate. In order to validate the analytic model, we compute the same metrics, overall and individual delivery times, given by the expressions in Tab. 1.

As a first evaluation, we assume that node interactions can be well described by the aggregate inter-contact time distributions, that is, the empirical distribution of inter-contact times estimated over all possible node pairs. From the aggregate distribution we find the contact rate as the reciprocal of average inter-contact time. Figs. 1 (b) and (c) depict the simulation results and the expected delivery times (denoted by **model**^{a}). Clearly, this method significantly underestimates the delivery delays, calling for more careful investigation of pair-wise node interactions. Therefore, we propose the second method to estimate the contact rate for a set of nodes. First, for every pair of nodes which reported contacts we find the average contact rate. Then, we find the empirical distributions of those contact rates and perform curve fitting. In all four cases, log-normal distribution seems to provide the best fit for the contact rate distributions. Next, we find the average contact rates from the fitted distributions and calculate the expected delivery times. Bars denoted by **model**^{b} in Fig. 1 correspond to this case. With a slight improvement from the previous method, the discrepancies between the analytic and simulation results are still significant. Acknowledging the fact that contacts for many node pairs are not observable from the traces (from 9% in the *Infocom* to almost 70% in the *Supsi* trace), we propose the third method to estimate average contact rate over all node pairs, by compensating for the missing node pairs. To all node pairs which have not recorded any contacts—the two nodes which haven't met during the experiment—we assign the same value for the average inter-contact time (for the delays in Fig. 1 that value equals the duration of the full trace), find the average contact rate over all pairs, and compute the delivery times. These results are denoted by **model**^{c}. We observe that this method gives better estimation than the previous two. However, the inconsistency (in some scenarios the method underestimates delivery delays, while in other vastly overestimates), makes it impracticable for use in general.

3. SUMMARY

We studied the performance of epidemic content distribution in opportunistic networks and empirically evaluated the content delivery delays by using four mobility datasets, chosen to represent a small system of users moving in a bounded area. We proposed three methods of treating the statistical data obtained from the traces, and showed that the homogeneous model is unable to accurately capture the epidemic process of the real-life scenarios. For our future work, we will aim at modeling epidemic spreading in heterogeneous systems by using other types of stochastic models.

4. REFERENCES

[1] J. M. Cabero, V. Molina, I. Urteaga, F. Liberal, and J. L. Martin. CRAWDAD data set Tecnalia Humanet (v. 2012-06-12), June 2012.

[2] A. Förster, K. Garg, H. A. Nguyen, and S. Giordano. On context awareness and social distance in human mobility traces. In *Proc.ACM*, MobiOpp '12, New York, NY, USA, 2012.

[3] O. Helgason, F. Legendre, V. Lenders, M. May, and G. Karlsson. Performance of opportunistic content distribution under different levels of cooperation. In *Proc. European Wireless Conference (EW)*, 2010.

[4] P. Meroni, S. Gaito, E. Pagani, and G. P. Rossi. CRAWDAD data set unimi/pmtr (v. 2008-12-01), Dec. 2008.

[5] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau. CRAWDAD trace cambridge/haggle/imote/infocom2006, May 2009.

Presentation of a Simulation Tool for Modeling Grid Stability in the Smart Grid

Florian Niedermeier, Gergő Lovász, Hermann de Meer and Thomas Koppera
Chair of Computer Networks and Communications
Innstr. 43
94032 Passau, Germany
Email: {niederme|lovasz|demeer|koppera}@fim.uni-passau.de

I. Introduction

The future Smart Grid will support the increased feed in of decentralized renewable energy sources and an intelligent management of grid entities. However, introducing these features brings certain challenges. First, increasing the fraction of renewable energy in the power grid requires power utilities to deal with the volatile nature of solar or wind power generation. This volatility may lead to grid instabilities, especially voltage stability [1] is a key concern in areas of high solar power generation. To cope with these challenges for grid stability, voltage irregularities have to be detected as early as possible. Second, intelligent management solutions like demand response [2] are required to be highly scalable to meet the demands of large power grids. However, as the power grid is a critical infrastructure which does not tolerate long downtimes, these demand response algorithms cannot be evaluated in actual field tests. Both challenges - a detailed grid monitoring and the possibility to evaluate new management solutions - introduce delays when implementing smart grid solutions. In case an environment does not allow for actual grid monitoring to be implemented or access to a testbed for testing smart grid algorithms does not exist, simulation may provide a feasible alternative.

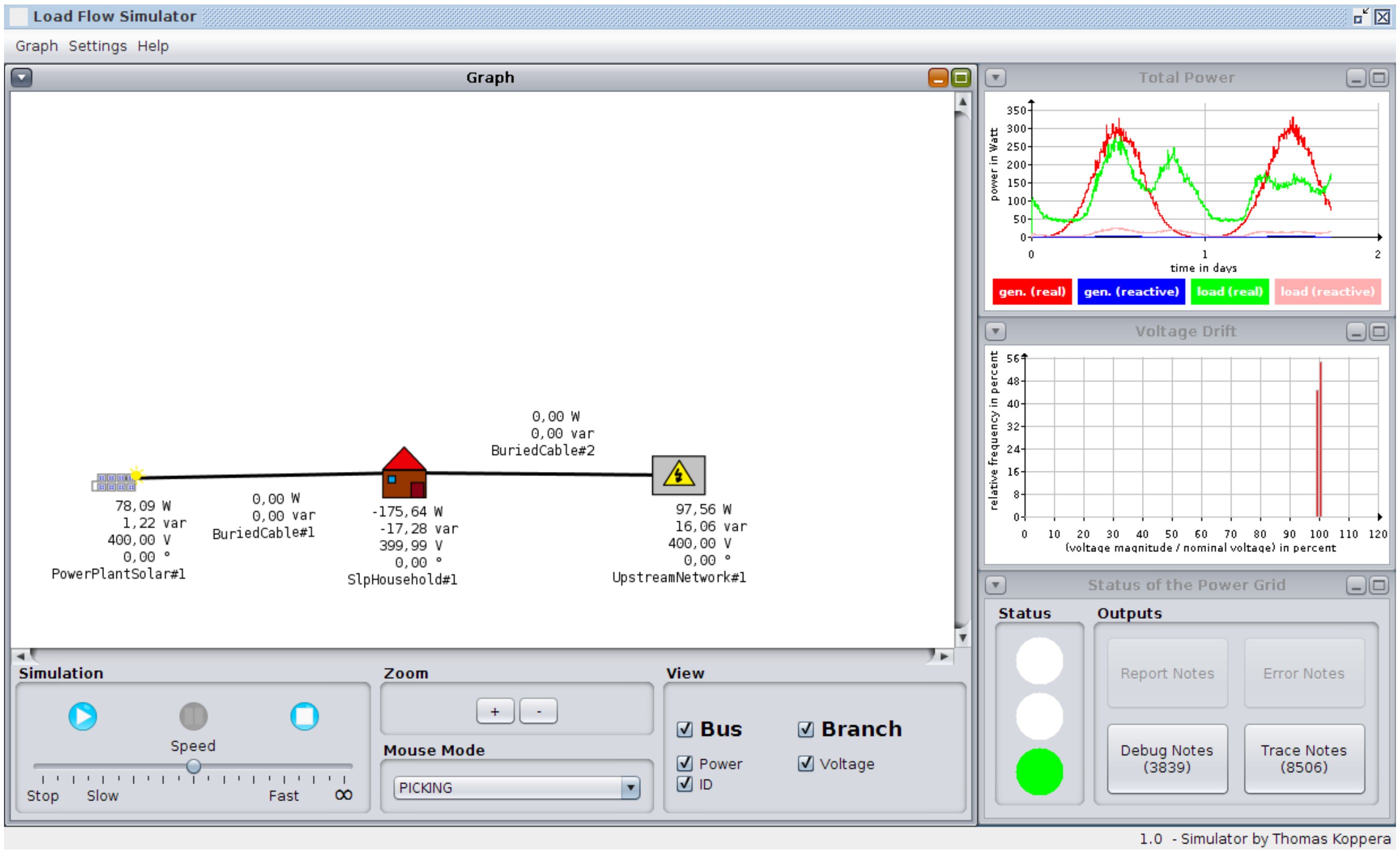
II. Simulation tool

In this abstract we present a work-in-progress power grid simulation tool which tackles both challenges. The tool is developed as open source software allowing the adaptation and extension of the tool. The tool allows modeling a power grid with different voltage levels via an user-friendly drag&drop interface, including fossil power generation, renewable generation and consumers [3]. The actual power grid simulation is performed using a discrete event simulation [4] in 15 minute steps. Inside the simulation the power grid is modeled as an undirected graph $G = (V, E)$. V denotes a set of vertices representing producers and consumers inside the power grid. It is noteworthy that a consumer can change its role to a producer at any time to account for prosumers in the actual power grid (e.g., household with high renewable energy feed-in). E denotes a set of edges connecting two vertices each ($E \in V \times V$). Edges represent power lines inside the power grid with certain properties such as maximum load, resistance, length, inductance, etc. One of the main applications of the tool is analyzing power grid stability. To this end it employs a Gauss-Seidel power flow algorithm. Another application is the simulation of demand response interactions. This is achieved by providing a generic interface which allows different concrete demand response algorithms to be easily integrated. Figure 1 depicts the graphical user interface (GUI) of the simulation tool. The largest area is used by a drag&drop capable window which depicts the topology of the simulated power grid. The user may interact with this window to easily construct power networks, including power plants, transformers, households and the connections in between. The upper right window shows a graph depicting the graphs of generation and demand of power, both for real and reactive power. The middle left panel shows a histogram of the relative voltage magnitude distribution. It is used to identify the amount and duration of voltage fluctuations, even in non-critical situations. The bottom right status window gives an overall status overview in form of traffic lights and signals if any simulated values (e.g., voltage) are outside the tolerable margins. The green light signals that the power grid is in a stable state whereas the orange light warns from an emerging instability in the power grid. The red light stands for a critical grid state that would lead to a blackout in the real power grid. The bottom middle and left panels give options regarding the simulation speed and view details. Topologies may be saved and opened later by using a XML-based file format [5]. The GUI of the simulation tool allows for an intuitive use of the tool. Especially the visualization of simulation runs and grid state allows to see easily how the change of certain parameters influences the power grid. If needed, the tool can easily be extended to support a scripted input interface since the simulation tool uses an XML-based representation of power grid models. If scripted input is preferred, the GUI could still be used to visualize simulation runs and information on grid state.

III. Related Work

There are several proprietary and open source simulation tools for analyzing power systems. Here, two characteristic examples of open source power analysis tools are briefly described: OpenDSS[1] and GridLAB-D[2]. Besides several methods for frequency domain analysis OpenDSS supports the research of smart grid and topics related to energy research. However, no graphical input and only limited output is provided by the tool. Furthermore, the multitude of functionalities supported by the tool requires an intensive training before the tool can be used. GridLAB-D is a freely available open source power distribution system simulation and analysis tool. Similarly to

[1]http://www.smartgrid.epri.com/SimulationTool.aspx
[2]http://www.gridlabd.org/



```
                                              0,00 W
                                              0,00 var
                                          BuriedCable#2

   78,09 W          0,00 W
    1,22 var        0,00 var      -175,64 W                         97,56 W
  400,00 V      BuriedCable#1      -17,28 var                       16,06 var
    0,00 °                         399,99 V                        400,00 V
PowerPlantSolar#1                    0,00 °                           0,00 °
                                  SlpHousehold#1                UpstreamNetwork#1
```

Mouse Mode: PICKING

View: ☑ Bus ☑ Power ☑ ID ☑ Branch ☑ Voltage

Debug Notes (3839)

Trace Notes (8506)

Fig. 1. The main GUI of the simulation tool

OpenDSS, it provides an extensive suite of analyzing tools. GridLAB-D allows also the modeling of emerging smart grid concepts such as demand response. The training material for GridLAB-D contains several hours of video courses that are necessary for getting familiar with the complex functionalities of the tool. Further simulation tools with similar characteristics are, e.g., the proprietary tools SINCAL[3] and PowerFactory[4]. In contrast to existing tools, our simulation tool offers only basic power analysis functionalities, but since it is open source and plug-in-based it can be easily extended by new functionality. The graphical user interface allows to start immediately with modeling without needing intensive training to understand the tool. Therefore, the tool is ideal to be used in student laboratory courses accompanying lectures.

References

[1] Deutsches Intstitut fr Normung-Din, *DIN EN 50160 Merkmale der Spannung in ffentlichen Elektrizittsversorgungsnetzen; Deutsche Fassung EN 50160:2010 + Cor. :2010.* Berlin Wien Zrich: Beuth Verlag, Februar 2011.

[2] S. von Roon and T. Gobmaier, "Demand response in der industrie–status und potenziale in deutschland," *München: Forschungsstelle für Energiewirtschaft eV (FfE)*, 2010.

[3] J. Glover, M. Sarma, and T. Overbye, *Power System Analysis and Design*. Cengage Learning, 2012.

[4] T. Lechler and B. Page, "Desmo-j: An object oriented discrete simulation framework in java," in *Proceedings of the 11th European Simulation Symposium*, 1999, pp. 46–50.

[5] U. Brandes, M. Eiglsperger, M. Kaufmann, J. Lerner, and C. Pich, "The graphml file format," 2000.

[3]http://www.energy.siemens.com/hq/en/services/power-transmission-distribution/power-technologies-international/software-solutions/pss-sincal.htm

[4]http://www.digsilent.de/index.php/products-powerfactory.html

EML, an Energy Measurement Library

Alberto Cabrera[*], Francisco Almeida[†] and Vicente Blanco[†]
[*] Instituto Tecnológico y de Energías Renovables. ITER
Granadilla, Tenerife. Spain
Email: acabrera@iter.es
[†] HPC Group. ETS de Ingeniería Informática
Universidad de La Laguna, ULL
La Laguna. 38270 Tenerife. Spain
Email: {falmeida, Vicente.Blanco}@ull.es

I. Introduction

Over the past years, energy aware computing has become a hot topic in High Performance Computing (HPC), as the work on the field towards the exascale computing has been progressing. To achieve exascale performance, several aspects of current computational models are being reviewed taking into account the energy consumption.

In the case of the algorithmic analysis, approaches like performance profiling and analytic modeling require good measurement tools to achieve the goals without having to deal with the concrete details of every hardware architecture and their respective challenges, such as how to properly measure energy.

Work in the field has been made to solve the problem of energy measurement by authors like Bellosa [1], Ryffel [2] and Ge et al. [3], but as the work was not directly available or it was a hardware solution, we started to develop a portable library called EML (Energy Measurement Library). With this implementation it is possible to abstract the user from the measuring tools, allowing to perform easiest measurement and, depending on the grade of precision and intrusion that we can have within the experiment, choose between external measurements or code instrumentation, as shown in Figure 1. Uses for this kind of library are decision making and auto-tuning based on energy consumption among other possibilities. To achieve so, we offer an unified interface that has a backend with implementeations for every measurement tool, which is easy to extend.

We currently have for measuring energy consumption the following implementations:

- PDU's
- Intel MSR
- A cluster with a measurement frontend

Figure 2 illustrates an example of measurement within the core using the Intel MSR Interface. It shows ten seconds of three obtained power profiles that belong to an execution of various matrix multiplications using the Intel MKL dgemm routine. Power profiles PP0 and PP1 belong to the processor while DRAM is the DIMM power consumption. Another example of measurement is Figure 3. In this case, measurement corresponds to an execution of High Performance Linpack, and

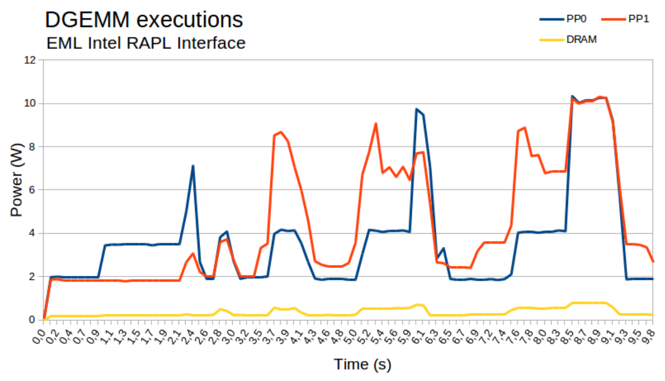


Fig. 2. Energy measurement from a processor using the Intel MSR RAPL interface.

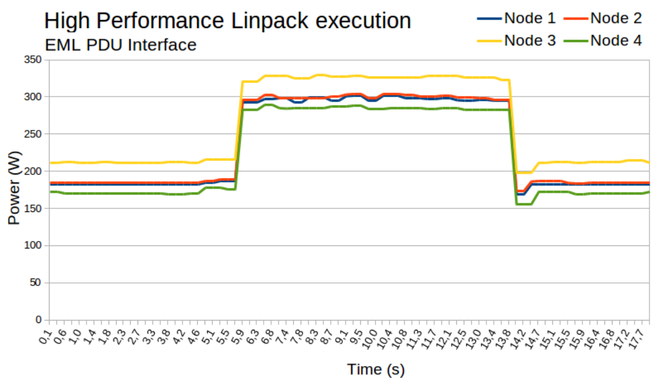
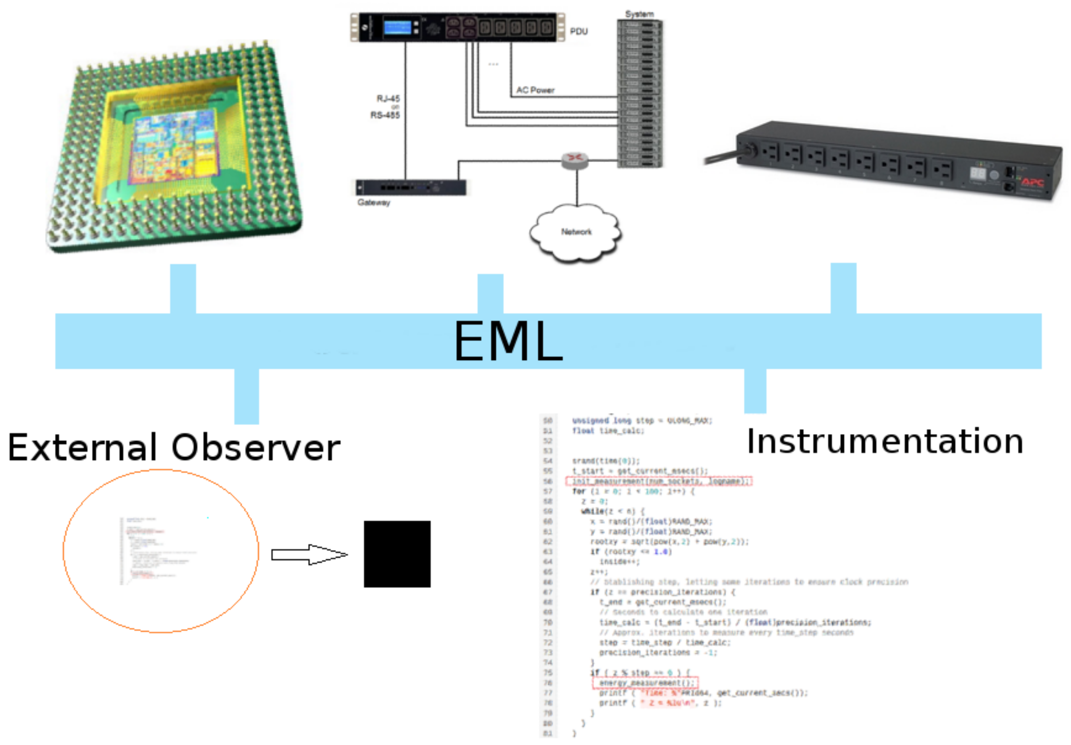


Fig. 3. Energy measurement using an external metered PDU from a monitor node.

is taken directly from pdu outlets using 4 nodes. Both outputs are logged in a file that can be formatted using timestamps to synchronize energy measurement and program execution. EML allows portability for our code in different architectures, provided that the new machine is compatible with at least one of the coded solutions.

II. Details

The current software design is thought to minimize the effort of adding code for new measurement tools. The simplest option is to structure the library as a Factory Method design



```
unsigned long step = ULONG_MAX;
float time_calc;


srand(time(0));
t_start = get_current_msecs();
init_measurement(num_sockets, logname);
for (i = 0; i < 100; i++) {
  z = 0;
  while(z < n) {
    x = rand()/(float)RAND_MAX;
    y = rand()/(float)RAND_MAX;
    rootxy = sqrt(pow(x,2) + pow(y,2));
    if (rootxy <= 1.0)
      inside++;
    z++;
    // Stablishing step, letting some iterations to ensure clock precision
    if (z == precision_iterations) {
      t_end = get_current_msecs();
      // Seconds to calculate one iteration
      time_calc = (t_end - t_start) / (float)precision_iterations;
      // Approx. iterations to measure every time_step seconds
      step = time_step / time_calc;
      precision_iterations = -1;
    }
    if ( z % step == 0 ) {
      energy_measurement();
      printf ( "Time: %"PRId64, get_current_secs());
      printf ( " Z = %lu\n", z );
    }
  }
}
```

Fig. 1. EML overview diagram. The upper part represents different hardware configurations (A processor that has Intel MSR RAPL, an external node to monitor a cluster or direct access to a PDU. The lower part shows the two different ways of adquiring data: an external program that measures energy consumption or code instrumentation. EML acts as a interconnection layer.

pattern, with every measurement tool represented as a class implementing two possible methods:

- *instant_measurement*. A method that returns the instant energy consumption measured by the hardware.
- *interval_measurement*. A method that returns the energy measured in an interval defined by the user. To determine the correct interval, this method is called following the classical time measurement: before and after performing the operation that requires measurement.

This involves a few challenges. First, the measurement tools we have encountered have one of the two methods directly available, so the second method has to be derived from the values obtained from the available one. From *instant_measurement* to *interval_measurement* the operation consists on measuring on the best possible interval given by the hardware constraints and then performing the correspondent integrating operation. The opposite, from *interval_measurement* to *instant_measurement*, is implemented in a similar way. Measurement is done on a given interval, then the instant value in watts is equal to the amount of energy resulting from resolving the equation

$$P = \frac{E}{s} \tag{1}$$

P is our instant power consumption, obtained thanks to the energy E measured every s seconds.

Our second challenge is to unify the different information obtained by the different energy measurement tools that could generate confusion. Given the concrete case of the Intel RAPL MSR and our metered PDU, the first obtains energy for a processor, separated in three different measures: the core itself, the RAM DIMMs and what Intel calls the *uncore* which, in Sandy Bridge platforms, measures the energy of the on–chip graphics processor. The latter returns the energy of a whole node. For now this is resolved by tagging the measurement taken. For future work, we pretend to extend EML capabilities to facilitate the analysis of the data offering different outputs such as XML.

III. Acknowledgments

This work was supported by the Spanish Ministry of Education and Science through the TIN2011-24598 project and the Spanish network CAPAP-H4.

References

[1] F. Bellosa, "The benefits of event: driven energy accounting in power-sensitive systems," in *ACM SIGOPS European Workshop*. ACM, 2000, pp. 37–42.
[2] S. Ryffel, "Lea^{2}p: The linux energy attribution and accounting platform," Master's thesis, Swiss Federal Institute of Technology, 2009.
[3] R. Ge, X. Feng, S. Song, H.-C. Chang, D. Li, and K. W. Cameron, "Powerpack: Energy profiling and analysis of high-performance systems and applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 5, pp. 658–671, 2010.