heterogeneous Multiprocessor Architecture

Houssam Eddine ZAHAF
Oran University
Computer Science departement
31000, Oran, ALgeria
Email: Houssam-Eddine.zahaf@univ-lille1.fr
Abou Elhassen BENYAMINA
Oran University
Computer Science departement
31000, Oran, Algeria
Email: benyanabou@yahoo.fr
Richard OLEJNIK
Lille1 UniversityLiFL
59650, villeneuve d'ascq, France
Email: Richard.olejnik@lifl.fr

Abstract—Analyse data issued from Social networks, large scale wireless networks, .. is computation intensive, and submitted to soft or hard real time constraints. The main characteristic of these kind of applications is that the execution time is greater than the deadline. Thus, mono processor architectures can not satisfy real time requirement of this type of applications. Multiprocessors nowadays architectures consists of numerous processors on one chip and allows to run tasks in parallel manner and can handle the overrun of these applications. In this paper, we present a novel online scheduler for real times tasks where execution time is greater than deadline. As application example, we use MapReduce Real time environments to extract simulation parameters and run tests on simS simulateur.

I.INTRODUCTION

Real time schedulers schedule tasks based on their real time charactiristics. Sporadic real time charactiristics are the Arrival time (R), deadline (D), least period of interactivation (P) , and worst case execution time WCET or C. In general, The relative deadline is greater than the execution time. However, Intensive applications deals with a huge amount of data and the WCET
is at least equal to deadline.
An important part of processing of an intesive computing application can be run in parrallel. That makes this applications more suitable to be run on multiprocessor architectures than on monoprocessors one. More than that, not any task set can be schedulable on one core architecture.
Most real time scheduling works focus on homogeneous MP- SoCs where all processors have the same speed and the same power consumption. However, Heterogeneous MPSoCs are more adiquate in terms of energy consumption and computing speed.
In this work, we focus on scheduling intensive real time tasks with energy constraints on uniform hardware architectures. The aim of the work is to decompose the real time task, to parrallel independant jobs with thier own real time charactiristics and we present our novel on line job-scheduler.
Unfortunately, schedulability test for heterogeneous are much harder, it depends not only on tasks, but on wich processor will run wich task.

II. BACKGROUND

In this work, we consider a set of n sporadic tasks on m processors. Each task is characterized by quadruple (R: Arrival Time, Period Between two activations: P, Deadline: D, WCET: Worst Case Execution Time). Each task is independant, and
have an implicit parrallelized sections.
First, we will present prior works and implementation of Map
Reduce Real time environments.

A. Taxonomy of multiprocessors

In terms of heterogeniety, MPSoCs can be classified as :

- Homogenoues

Each task or job is run at the same speed on each processor and consumes the same energy.

- Unifrom

Processors may habe different speeds, but a task that runs in 3 time units on a processor with speed 1, run in 1.5 time units on a processor with speed 2, and 0.75 on processor of speed 4. Each processors consumes at least quadratic of speed on energy compared to a processor with speed 1.

- Unrelated heterogenous

The execution time and the consumed energy depends on the task and the processor at the same time.
In this work we will focus on uniform architectures only.

B. Map Reduce Real Time environments

First, we will provide an overview on Map Reduce and its open source implementation Hadoop, and we focus after on Real-time Map Reduce environments, Exactly Hadoop real time implementations and Misco RT. We will discuss them strength points and weaknesses. A. MapReduce MapReduce [1] is parallelized, distributed platform for large scale data processing. It virtualizes task and data mapping and scheduling, communication, running failure, fault tolerance and all execution details.
Map Reduce is quite simple, it split a big computing task [10] to smaller ones, each sub-task is affected a worker node. These splits are independent and each worker lunches a different piece of input data. Task independence allows running tasks in parallel manor and the re-run possibility for fault tolerance. User defines only two functions Map and Reduce. Map Task
is applied on a set of input data and produce ¡Key, values¿, the second function reduce allows to reduce partial results and

producing final ones.

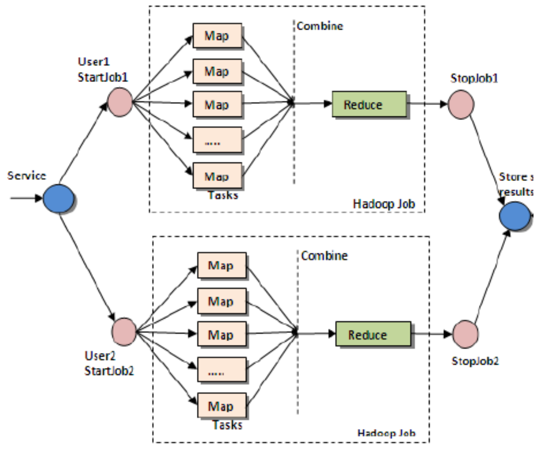

Fig. 1. Map Reduce

1) Hadoop: Apache Hadoop is an open-source Map Re- duce framework on clusters of commodity hardware. Hadoop
is an Apache top-level project being built and used by a global community of contributors and users.

The Apache Hadoop framework is composed of the fol- lowing modules:

• Hadoop Common contains libraries and utilities needed by other Hadoop modules.

• Hadoop Distributed File System (HDFS) a chunk based dis

• Hadoop YARN a resource-management platform re- sponsible for managing compute resources in clusters and using them for mapping and scheduling of users applications.[12]

Java is the used language with ”Hadoop Streaming” to implement the ”map” and ”reduce” parts of the user’s program. The Hadoop framework itself is mostly written in the Java, with some native code in C and command line utilities written as shell-scripts.

2) Hadoop Scheduler: Hadoop job schedulers are FIFO, and fair scheduler, not like FIFO scheduler fig 3, In fair scheduling, tasks not is the top of the queue, may be scheduled by assigning tasks into different pools, and assign to each pool, minimum guaranteed share. Figure 2 and
3 show the difference between both of FIFO and fair scheduler.

Each pool is characterized by the number of Map and
Reduce slots and the number of the maximum jobs assigned.

The scheduling algorithm is simple; first, it splits each pools min share among its jobs and split each pools total share among its jobs. When a slot needs to be assigned: If there is any job below its min share, schedule it. Else schedule

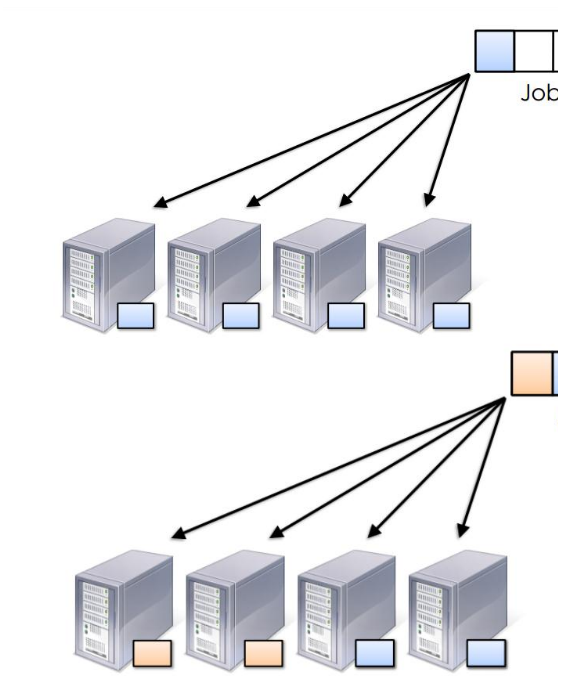

Fig. 2. Fifo and Fair scheduler

the job that has been most unfair to (based on deficit).Phan
et al in [6] explored the feasibility of enabling scheduling of mixed hard and soft real time map reduce applications. They tried to investigate the impact of some factors over the respect of time constraints such as data placement, concurrent users, and communication bandwidth.
The aim of the work of [6] is to use already existed scheduling real time algorithms on EC2 Amazon cloud. The tried to provide a scheduling algorithm to insure that hard real time tasks meet their deadline and try to satisfy soft real time constraints or at least minimize tardiness.
They focus on three points, what can affect the real time scheduling, based on results of the first investigation; they formulated the problem like a Constraint Satisfaction Problem CSP, the third step was solving the problem with a new heuristic for real time MapReduce tasks scheduling.

First they define parameters influencing real time schedul- ing as the number of map and reduce slots per a cote, multiple concurrent jobs, data placement, the interval of heart beats, and the algorithm of scheduling itself. Scheduling problem was formulated to a set of real MapReduce applications on
a distributed heterogeneous architecture as a CSP, which can be solved using well-known constraint solvers. They focused first on the offline setting, where the set of Map Reduce jobs are known a priori and the role of the scheduler is then to determine an optimal execution schedule for all tasks. The novelty of [] is that the formulation lies in the modeling of various factors unique to the Map Reduce jobs. Specifically, formulation considers slot-to-core ratio, the effect of input data placement on the data transfer time (from a remote or
a local host), and the interval based on heart beats between the master and the slaves. The last factor is the heterogeneity of the processors, where a tasks execution time varies based on the slots processing capability. The CSP formulation is based on very restrictive assumptions like:

each job contains no more than one map stage and one reduce stage. the worst-case execution time (WCET) of a task on each processor type is known a priori; all processors work perfectly without failure; and there is no speculative execution and no task migration.

The WCET assumption is necessary for real-time, they used WCET evaluator described in[9]. Gecode is the solver used to implement the formulated CSP. We can denote the absence of energy in the formulation, not only that but also the processor failure, which is a common thing in embedded hardware architectures. Using a solver is a high cost for problems like scheduling and mapping.

C. Hadoop for soft Real Time

Dong X et al in [1] have proposed an adaptation of Hadoop scheduler to support real time constraints, it allows to schedule mixed real time and no real time applications. The main contributions of this work are task forward scheduler and resources allocation model. The scheduler is compound of three sub-schedulers, Real time scheduler called deadline scheduler, no real time scheduler, and master scheduler that combine both. Thus allows reusing no real time schedulers. To grant real time constraints, they had used on line execution time evaluator proposed in [11]. One real time map and reduce job are picked randomly and submitted to sampling phase. The results of sampling phase are execution time for map and reduce task. The approximated calculated execution time is divided and added to correction values. Real time and no real time applications are queued equally, with a higher priority for real time applications.

Queued tasks are managed by deadline scheduler, and an existing no real time scheduler. If real time tasks cannot get needed resources, they can preempt some from non-real time tasks. In [1], The system assume that the execution platform is homogeneous and all input data have the same size. The aim of the deadline scheduler is to maximize concurrent real time jobs on the minimum number of resources, and determine if scheduling a new task is feasible or not without influencing already scheduled tasks. Resource allocation modal aim to define the minimum size of parallelized real time jobs in order to maximize the number of concurrent real time jobs. Each cluster is composed by Map nodes, reduce nodes, real time map nodes, and real-time reduce nodes. A scheduled task assign query, by arrival time, relative deadline, the number of map and reduce tasks, execution time for each map and reduce task. The second step is defining the correct work load for each worker. It tries to make jobs as small as possible so the number of concurrent real time jobs is maximized. According to [1] defining the Degree of parallelism (work load) is optimal, and all the concurrent jobs have approximately the same size of by time unit. Jobs have the same size, execution platform homogeneous, the DOP is equal for all tasks, all these can be noticed as drawbacks and as restriction for this work. More, energy consumption and data load and delivery time impact is ignored.

D. Misco RT

Misco RT is a python implementation of Soft real-time Map Reduce. Unlike works discussed earlier, Misco RT runs on homogeneous embedded architecture architectures and it considers processor failures.

Misco architecture is simple, It comprises a Master Server and a set of Worker nodes. Server maps and schedules tasks on worker nodes, and keeps the execution tracks. The worker node is able to run even a Map and Reduce task. Each task
is characterized by the father application and the location of data (one input file), ready time, and real time characteris- tics. The Misco system is considered as a set of distributed applications A1, A2, .., An, compound of a set of Map and Reduce Tasks, applications are sporadic and their arrival time
is unknown a priori witch makes the system less predictable. The Master Server keeps track of user applications, while the Worker Nodes are responsible for performing the map and reduce operations. The Misco server also maintains the input, intermediary and result data associated with the applications, keeps track of their progress and determines how application tasks should be assigned to workers.

The main responsibility for the Misco worker is to process the individual map and reduce tasks and return the results to the server. The Misco worker consists of a Requester component,
a Task Repository component and a Logger component. The Requester is used for interactions with the Misco server to request tasks and download and upload data, trigger the local execution of the tasks, and handle the communication with the Misco system during upgrades. The Misco server is in charge of keeping track of applications submitted by the user and assigning tasks to workers. It comprises Scheduler that implements our two-level scheduling scheme, an Application Repository that keeps track of application input and output data, and an HTTP Server that serves as the main communi- cation between the workers and the Misco server.

Real time characteristics are defined by the user, and the execution time of an application depend on Map and reduce times and data load and delivery time. Misco RT applications and tasks scheduler is based on LLF, each calculated laxity is considered as its urgency. The main goal is not to reducing delays but maximizing the number of applications meeting their deadlines. However, this can cause a running failure because a dropped task will probably mean that all executed tasks of the same application was in vain.

Worker in Misco failure is permanent or transient. When a worker fails, all assigned tasks are lost. Server computes failure rate and redistributes failed tasks. Each free worker sends a request for jobs. Misco RT platform runs only on homogeneous hardware architectures, Misco platform is too much restricted, and run only on homogeneous hardware architecture, we denote also the absence of energy consumption especially for an embedded system like mobile phone, the reason for the platform was ever developed, we can notice the absence of a mechanism that insure the real time data transfer.

Real time applications are almost critical and data sensitive, however, few works had focus on the security aspect in map reduce environment, Roy et al in [12] have proposed a MapReduce environment with an enhanced security. Airavat
is a MapReduce-based system which provides strong security
and privacy guarantees for distributed computations on sensi- tive data. Airavat is a novel integration of mandatory access control and differential privacy. Data providers control the security policy for their sensitive data. Airavat confines these computations, preventing information leakage beyond the data providers policy. Airavat is modular and can be integrated to any map reduce platform. The prototype is efficient. Airavat will not be discussed in this paper just mentioned like a possible feature to complete limits of security for other Real time map reduce works.

III. CONTRIBUTION

The goal of our design is to provide more deterministic real time treatment and grant low energy consumption by exploiting the heterogeneity of the hardware architecture. Our system design is multilayer system, compound from 4 levels (fig 4)

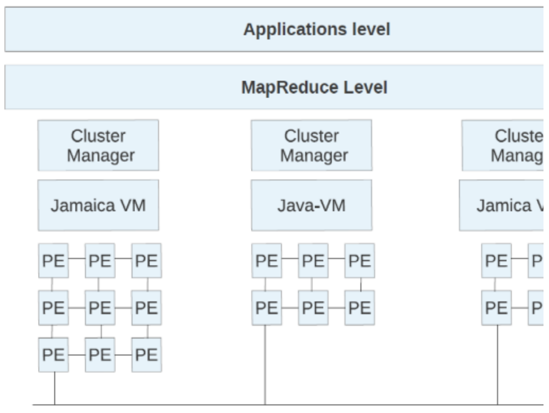

Fig. 3. Our design

Fig4. System design architecture

A. Application layer

In this layer the user defines each application by defining its map and reduce task with the real time characteristic (Type: Periodic or sporadic, Offset, Arrival Time, deadline) and data size, data location, splitible parameter witch define if a task can be spliced into subtasks to be run on parallel or not. MRID
is an integer that expresses the precedence order of a set of map tasks and one reduce task, semantically it means that the reduce task will reduce the results of that map tasks. And its defined automatically by the system. B. MapReduce Layer: It contains our main contributions. It consists of 5 entities, Real-time scheduler, requester, Data Splitter, Entities connection are shown in fig 6.

B. Scheduler

Applications scheduler is a fair scheduler with a maximum share per queue. Task scheduler is EDF (earliest deadline First) based scheduler, both of map and reduce tasks are sorted by deadline. In an obvious manor, Reduce tasks will be placed after the correspondent Map tasks (its deadline equals the deadline of the last map task plus its execution time). The second consist to give reduce tasks a better scheduling, because running the reduce task, mean concluding a set of processing,

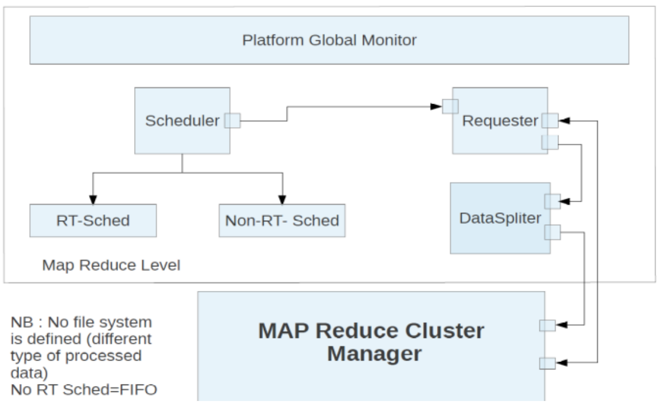

Fig. 4. Component connections

so free processors and memory. Each Reduce tasks then are higher priority than every Map task that does not equal to its MRID. Ready time for a Reduce task is computed, and not given by the user; it equals the last execution time for the last correspondent Final map task (see section data splitter).

C. Requester

It takes the scheduled task and sends a request to each processor in order to get free fits that corresponds between the arrival time and deadline of the selected task. It sends arrival time, deadline, and data size to each worker node. These lasts must reply by the executing fit. the execution rate, and the energy consumed if the map task will occupy that free fit. Each reply will be : (Processor, fit, execution rate, energy consumed). All replies will be sent to data splitter. The evaluation of the energy consumed and the execution rate will be discussed in section worker

D. Data Splitter

Compound from two levels, Solver and launcher. It takes the results of requester and tries to find the best solution that grant the execution of the task and the low consumption. The formulation of that problem is defined in formula 1.

A reasonable solution is a solution where the sum of rates equals or higher than 1. If a fit is considered then $x_i = 1$ else
$= 0$. To exclude over running values we introduce the objective function min P. Minimize the energy consumed is the most important objective. The energy consumed depends on several parameters discussed in section worker.

$$MinE = \sum e_i * x_i$$

$$\mathrm{MinP} = \mathbf{P}\; p_i * x_i$$

$$\mathbf{P}\; p_i * x_i \geq 1$$

To solve this problem, we use a branch and bound resolution method.

$$V \qquad + P_e * T_l * S_d$$

E. Worker

The worker is able to run both of map and reduce tasks, It contains three main modules, Load

$$P = \quad e \quad \text{————————} \; V * S_f$$

$$WCET + T_l * S_d * V$$

$$E = P_e * v^2 * S_d$$

The Worst case load time and worst case execution time are not so deterministic values, so the monitors scale the computing speed to take benifit for unused inter fit space.

For simulation, we use SimSo simulator, It is an simulator for real time on multiprocessors uniform hardware architecture. Our approach stills in tests phases, and the final results will be published as soon as possible

IV. CONCLUSIONS

Our design can be easily plugged to Misco RT, or Hadoop, its modular and simple, however our design is for embedded hardware architecture like MPSoCs Based NoC, Hadoop is too much computational for such hardware architecture, Misco
is too much restricted, so we implemented our Map Reduce environment mainly in java with some native C code.

REFERENCES

[1] Dong, X., Wang, Y., Liao, H. (2011, December). Scheduling mixed real- time and non-real-time applications in mapreduce environment. In Par- allel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on(pp. 9-16). IEEE.

[2] Dou, A., Kalogeraki, V., Gunopulos, D. Mielikainen, T. Tuulos, V. H. (2010, June). Misco: a MapReduce framework for mobile systems. In Proceedings of the 3rd international conference on pervasive technologies related to assistive environments (p. 32). ACM.

[3] Dou, A. J., Kalogeraki, V., Gunopulos, D., Mielikainen, T., Tuulos, V. (2011, July). Scheduling for real-time mobile MapReduce systems. In Proceedings of the 5th ACM international conference on Distributed event-based system

[4] Mishra, R., Rastogi, N., Zhu, D., Moss, D., Melhem, R. (2003, April).

Evaluator, runner, and log
register. a) Load Evaluator : It calculates the execution rate P and the correspondent consumed energy E based on data load time, data size, Worst case execution time, data delivery time. What we can execute in a fit equals:

$$S_f = P_e * \frac{WCET}{}$$

[12] Lam, C.(2010). Hadoop in action, Manning Publications Co.

Energy aware scheduling for distributed real-time systems. In Parallel and
Distributed Processing Symposium, 2003. Proceedings. International (pp.
9-pp). IEEE.

[5] Dou, A. J., Kalogeraki, V., Gunopulos, D., Mielikainen, T., Tuulos, V. (2011, July). Scheduling for real-time mobile MapReduce systems. In Proceedings of the 5th ACM international conference on Distributed event-based system (pp. 347-358). ACM.

[6] Phan, L. T., Zhang, Z., Loo, B. T., Lee, I. (2010). Real-time MapReduce scheduling.

[7] Elespuru, P. R., Shakya, S., Mishra, S. (2009). MapReduce system over heterogeneous mobile devices. In Software technologies for embedded and ubiquitous systems (pp. 168-179). Springer Berlin Heidelberg.

[8] Owens, J. D., Houston, M., Luebke, D., Green, S., Stone, J. E., Phillips,
J. C. (2008). GPU computing. Proceedings of the IEEE, 96(5), 879-899.

[9] Wilhelm, R., Engblom, J., Ermedahl, A., Holsti, N., Thesing, S., Whalley, D., Stenstrm, P. (2008). The worst-case execution-time problemoverview of methods and survey of tools. ACM Transactions on Embedded Computing Systems (TECS), 7(3), 36.

[10] Polo, J., Castillo, C., Carrera, D., Becerra, Y., Whalley, I., Steinder, M., Ayguad, E. (2011). Resource-aware adaptive scheduling for MapReduce clusters. In Middleware 2011 (pp. 187-207). Springer Berlin Heidelberg

[11] Roy, I, Setty, S (2010, April). Airavat: Security and privacy for
MapReduce . In NSDI (Vol. 10,pp. 297-312).