Mathematical Problems of Computer Science 36, 121--127, 2012.

# Workload Management for Grid Environment with the Restriction on the Waiting Time

Vladimir Sahakyan and Sergey Petrosyan

Institute for Informatics and Automation Problems of NAS of RA

vladimir.sahakyan@sci.am sergpet@ipia.sci.am

**Abstract:**

Resource management and job scheduling in multiprocessor computing system and Grid environment are challenging problems. Although significant results were achieved in the past, there are some problems that still exist, and need to be completely solved. More restrictions in job will make queue management run efficiently. One of the main parameters in the job scheduling is a *waiting time*. Waiting time is the time period, that job is ready to wait until it runs.

In this article one approach to organize workload management is considered, it gives an overall solution for problem, and may be upgraded to support non-homogeneous systems or adding some new fetchers. The article offers two parametric models of queue service discipline FIFO with optimizations and restriction on waiting time. The models will be compared and underlined within the main usage of two models.

**Keywords:** Workload management, Queuing theory, Multiprocessor system, High performance computing system, Computing Grid.

## 1. Introduction

The aim of the ArmGrid [1] founded in Armenia is to build on recent advances Grid technology and develop a service Grid infrastructure available to scientists. This also means providing robust middleware components, deployable on several platforms and operating system, corresponding to a set of core Grid services, such as physics, geophysics, astronomy and bioinformatics that are utilizing grids to share, manage and process large data sets.

Workload management is one of these key Grid services. Asignificant result has been achieved on the problem of scheduling and efficiently managing a big number of different types of Grid infrastructure [2,3,4,5,6]. However the problem of Grid scheduling and resource management can of course not be considered as completely solved yet since many areas still require attention.

Taking into account the previous experience from other Grid projects, such feedback and requirements coming from the reference applications, the new queue organization mechanismis offered and presented in the rest of this paper.

In the rest of paper we will define job by these tree attributes$(\beta, \vartheta, \omega)$, the symbol $\vdash$ replace "if", the symbol $\Rightarrow$ replace "then". With bracket $\left\lfloor \frac{A}{B} \right\rfloor$ we will define the real part of whole $\frac{A}{B}$ number.

## 2. Definition

Let us discuss a distributed multiprocessor computing system which consists of K (K=1,2, ...) multiprocessor computing subsystems. The $i$ system we will call $M_i$ and the resource that has $M_i$ system we will denote by $R_i$. In this article $R_i$ has numerical value and shows the number of processors that system $M_i$has, but in general $R$may show more information about $M_i$ system.

Each job will be described by 3 main attributes: time that job needs to run$\beta$, the number of processors to run job$\vartheta$, and delay time $\omega$, time that job may wait until it runs. In the rest of paper we will define job by these tree attributes $(\beta, \vartheta, \omega)$.

## 3. Architecture of the Workload Managements System

The Workload Management System (WMS) comprises a set of Grid middleware components responsible for the distribution and management of tasks across Grid resources, in such a way that applications are conveniently, efficiently and effectively executed.
The specific kind of tasks that request computations are usually referred to "jobs". In the WMS, the scope of tasks needs to be broadened to take into account other kinds of resources, such as network, memory or processor capacity.

### *Functionality*

The core components for WMS are "Resource Selector" and "Resource Checker" that uses "Resource Info"; "Job Bookkeeper"; "Waiting Queue" and "Running Queue" which are Informational queues and lists. The purpose of core components is to accept and satisfy requests for job management, expressed via Job Description Language (JDL) [7] coming from its clients. If the computing system is busy the core components should understand that system does not have enough free resource to handle that job and reject it.

For a computation job there are two main types of request: submission and cancellation (the status request is managed by the "Resource Selector" and "Job bookkeeper").

In particular the meaning of the submission request is to pass the responsibility of the job from "Controller" to the "Resource Selector" which will find an appropriate sorted resource list in which the job will run. The "Resource Checker" will check if that resource is free to handle one new job, if it is so, then calculating the start and end times and add job either in "Running Queue" or in "Waiting Queue" making records in "Job Bookkeeper". If the selected resource does not have any free time to run this job, then "Resource Checker" starts to check next resource from list, until it finds appropriate resource or understand that there are no any free time in selected resource to serve this job, then system will reject the job with error that "There is no enough free resources".

The job cancelation starts modification in job bookkeeper and deletes records from "Running Queue" or "Waiting Queue".

After completing each request the system will call an appropriate procedure to run task in the computing element (CE).

Each component will be described below. All core components are independent modules and each module may be developed for a better result.

## 4. Core Components of WMS

Below all core components of workload managements system will be described, they are independent modules and each module may be developed for better result.

### *"Resource Selector"*

Each Job that is submitted into the system, comes to "Resource Selector" that selects the appropriate resource for new job. A "Resource Selector" can adopt eager or lazy policy in order to schedule a job. At one extreme, eager scheduling dictates that a job is bound to a resource as soon as possible and, once the decision has been taken, the job is passed to the selected resource for execution, where, very likely, it will end up in a queue. At the other extreme, lazy scheduling foresees that the job is held by the system until a resource becomes available, at which point that resource is matched against the submitted jobs and the job that fits best is passed to the resource for immediate execution. Varying degrees of eagerness (or laziness) are applicable.

At match-making level the main difference between the two extremes is that eager scheduling implies matching a job against multiple resources, whereas lazy scheduling implies matching a resource against multiple jobs.

### *"Resource Checker"*

"Resource Checker" starts, when an appropriate resource list is selected, and checks all lists starting first resource if the resource has enough free time to serve the arrived Job. If the resource that has enough time to serve arrived job is selected then "Resource Checker" sends the selected resource to "Job Submitter" and relative time when job will run in system. If no resource that can handle this job, then system will throw out the job with error code that there are no free resources to run this job.

## 5. First Approach to WMS organization (sequential)

In this module one approach to WMS construction is offered. As it mentioned in last module the core components are "Resource Selector" and "Resource Checker", it means that it is enough to construct those two components for WMS construction. The approach name is sequential because it starts to serve each job consecutively.

### *"Resource Selector"*

The model that is implemented in this WMS is FIFO model; it means that any job arriving first will run first, if the system can handle it. When Job arrives the "Resource Selector" selects the sorted by processor ascending resource list from "Resource Info" which has can handle arrived Job (for $(\beta_i, \vartheta_i, \omega_i)$ the result of "Resource Selector" is $\{M_h\}$ array where $\forall M_j \in \{M_h\}$, $\vartheta_i \le R_j$ , $\forall M_{j_1}, M_{j_2} \in \{M_h\}$, $\vdash j_1 < j_2 \Rightarrow R_{j_1} \le R_{j_2} \forall l, 1 \le l \le n \vdash M_l \notin \{M_h\} \Rightarrow R_l < \vartheta_i$).

If the $\{M_h\}$ array is empty the system will throw out the job with error code that there is no resource that can handle arrived job otherwise the system will start to check selected resource.

"Resource Selector" may be developed to support heterogeneous resource as well as different size of RAMs and network type resource and requirements.

### *"Resource Checker"*

"Resource Checker" calculates the $d_j(t_i + 0)$ and $v_j(t_i + 0)$, where $d_j(t_i + 0)$ is the list of jobs running in $M_j$ selected resource in current $t_i$ time and $v_j(t_i + 0)$ is the list of jobs waiting for run in $M_j$ selected resource in current $t_i$ time. After calculating$\tau_i$ that shows relative time, at which$(\beta_i, \vartheta_i, \omega_i)$ job may start to run in $M_j$ system $(\sum_{\vartheta_{jl} \in d_j(t_i+\tau_i)} \vartheta_{jl} \le R_j - \vartheta_i, \tau_i < \vartheta_i$and $\forall \tau_i' \in [0, \tau_i] \vdash \sum_{\vartheta_{jl} \in d_j(\tau)} \vartheta_{jl} \le R_j - \vartheta_i \Rightarrow \tau_i' = \tau_i)$. After "Resource Checker" starts to check starting from $\tau_i$if the system may handle $(\beta_i, \vartheta_i, \omega_i)$ in $\beta_i$ relative time (check if there is $\hat{\tau} \vdash \exists \hat{\tau} \in (\tau_i, \tau_i + \beta_i), \sum_{\vartheta_{jl} \in d_j(\tau_i)} \vartheta_{jl} > R_j - \vartheta_i)$. If there is a relative time when system may not handle job then "Resource Checker" to starts calculate next $\tau_i$ that shows relative time, at which$(\beta_i, \vartheta_i, \omega_i)$ job may be started in $M_j$ system and continues with the same algorithm until it finds an appropriate resource and start time for job.

The algorithm complexity is cubic, $O(n^3)$ where n for each job is the max($v_j(t_i +$ 0), $d_j(t_i + 0)$, K). Really, there are 3 types of cycles in the algorithm.For each computing system and each jobthe algorithm checks the waiting queue and running queue. The cycles are embedded in each other, and there is no other cycle in algorithm, then the complexity is cubic.

## 6. Second Approach to WMS organization (eager)

The second approach of WMS is similar to eager algorithm. For each job, in "Resource Checker", the algorithm checks, whether the job requires more or less resources comparing with resources that have computing system. If the resource requirement is less than some value and overall computing system is busy so the task will not be served by system. In any other case the system will run job if there is enough resources.

The algorithm for second approach is generally the same that is in the first approach. Only difference is the first part of "Resource Checker". Bellow will be described the "Resource Checker" for second approach.

### *"Resource Checker"*

For this approach "Resource Checker" at first calls "C" procedure with $t_i, (\beta_i, \vartheta_i, \omega_i), M_j$ arguments. If the result of "C" procedure is true, the "Resource Checker" will continue with the same algorithm as in first approach. If the result is false, then "Resource Checker" will not serve the job. The description of "C" procedure mentioned below.

$$C\left(t_i, (\beta_i, \vartheta_i, \omega_i), M_j\right) = \begin{cases} True \vdash \dfrac{\vartheta_i}{R_j} \ge K1 \\ True \vdash \dfrac{\vartheta_i}{R_j} < K1, \quad D(0, \omega_i + \beta_i) \le K2 \\ False otherwise \end{cases}$$

In this system K1 and K2 are coefficients that give Grid administrator.

The algorithm complexity is cubic, $O(n^3)$ because the deepest cycle in this algorithm is the same as in previse algorithm.

## 7. Sequential and Eager Algorithm Comparison

The workload management is similar to loading problem [4]. One of the approaches to loading problem solution is eager algorithm, but eager algorithm is not a good solution for that problem [8]. That is why it may cause that eager algorithm may be bad solution for WMS too. In the rest of paper some preconditions will be shown, and will be proofed the thesis that the usageof eager algorithm with preconditions will bring more resource utilization than the usage of sequential algorithm.

Let K grid system with $M_1, M_2, \dots M_n$ subsystems$(n = 1,2, \dots)$ is given. Each $M_i$ subsystem has $R_i$ resources. The $m$ value is the maximum of $R_i$resources $m = \max_{1 \le i \le n} R_i$. The $A = (a_{ij})$ is the matrix, where $1 \le i \le n$, $1 \le j \le m$ and $a_{ij} = \frac{R_i}{j}$. Let's define vectors$\bar{b} = (R_1, R_2, \dots R_n)$ and $\bar{c} = (c_1, c_2, \dots c_m)$. The definition of vector$c$ is:

$$c_j = \begin{cases} \frac{1}{m * K_1 - j} \vdash j < m * K_1 \\ 1 \vdash m * K_1 \le j \le m * K_2 \\ \frac{1}{j - m * K_2} \vdash j > m * K_2 \end{cases}$$

The vector$\bar{x} = (x_1, x_2, \dots x_m)$ is the variable vector. (1) Is the linear programming minimization problem.

$$\bar{x} \ge 0, A\bar{x} \le \bar{b}$$

$$\bar{x}\bar{c} \to \max \qquad (1)$$

Having solution of (1) problem it is easy to calculate$\varepsilon = \frac{1}{\sum_1^m x_j}$.

Let's define the set $\Omega = \{1,2, \dots, m\}$ and the subset $A = \{[m * K_1], [m * K_1] + 1, \dots, [m * (1 - K_2)]\}$. $p(j) = x_i * \varepsilon, 1 \le j \le m. P(A) = \sum_{i \in A} p(i)$. (2)

**Theorem:** If probability of arrived task into grid computing system is similar to (2) probability, $P(A) > 0.5$ and $2 * K_1 > (1 - K_2)$ then during long-term working the eager algorithm utilize more resources than sequential algorithm.

**Proof:** Let $Job_1, Job_2, \dots, Job_k, \dots$ are the jobs that comes into the system, and $A_1, A_2, \dots, A_k, \dots$; $B_1, B_2, \dots, B_k, \dots$ are the system status after sequential and eager algorithm accordingly. In $t = 0$ time the system is empty, that is why $A_1 = B_1$, $A_2 = B_2$ ... till system's overload, when $A_i \ne B_i$. It means that sequential algorithm put $Job_i$ into $M_e$ subsystem's queue, while eager algorithm reject $Job_i$. The eager algorithm rejects job if $C$ procedure gives "false" value, so it means $\frac{\vartheta_i}{R_f} < K1$ and $(0, \omega_i + \beta_i) \le K2(e \ge f)$. So $\vartheta_i < K_1 * R_f$ and $\vartheta(\beta_i + \omega_i) < K_2 * m * (\beta_i + \omega_i)$:

Let $A_1 = \{1, 2, \dots [m * (1 - K_2) - \vartheta_i]\}$ and $A_2 = \{[m * (1 - K_2) - \vartheta_i] + 1, [m * (1 - K_2) - \vartheta_i] + 2, \dots, [m * 1 - K_2)]\}$. $P(A_1)$ is the likelihood that next job will be served by computing subsystem, and$P(A_2)$ is the likelihood that after acceptation $Job_i$ the subsystem wouldn't serve next job. If $P(A_1) < P(A_2)$ (3) will be satisfied then the theorem will be proofed.

To satisfy (3) condition must meet $[m * (1 - K_2) + \vartheta_i] \leq [m * K_1]$. The condition $m * (1 - K_2) - \vartheta_i \leq m * K_1$ may be divided to $m$ because $m > 0$. It is easy to reform $(1 - K_2) - \vartheta_i/m \leq K_1$ condition into $(1 - K_2) - K_1 \leq \vartheta_i/m$, where $K_1$ could be replaced with $(1 - K_2)/2$ from preconditions. It is easy to see from $(1 - K_2)/2 \leq \frac{\vartheta_i}{m}$ that for satisfy (3) condition the $\vartheta_i$ must be large as possible but less then $[m * K_1]$. So it is easy to see from (1) LP problem solution that likelihood of $\vartheta_i$ is beiger then 1 is more, then it near 1. That is why in more situation (3) condition is right. $P(A_1) + P(A_2) \leq 1$ and $A \subseteq A_2$ then $P(A_1) < P(A)$ and $P(A_1) < 0.5 < P(A_2)$ (4). The (4) condition says that $Job_{i+1}$ that require more resource than $Job_i$ will be served by system only if system rejects $Job_i$ job. So the theorem is proofed.

## 8. Conclusion

As the result we have software, which uses mentioned methods for queue creation and management.

## References

[1] *Official web-site of the ArmenianNational Grid Initiative Foundation, http://www.grid.am*
[2] V. Sahakyan and S. Petrosyan, " Simulation of the queue with the restriction on the waiting time for multiprocessor systems ", *Proceedings of Conference Computer Science and Information Technologies*, pp. 272-273, 2011.
[3] V.Sahakyan, "About the queue organization in the multiprocessor computing systems", *Mathematical Problems of Computer Science*, vol. 34, pp.18-19, 2010.
[4] S. Petrosyan, "Simulation of the queue with the restriction on the waiting time for multiprocessor systems", *Proceedings of Conference Computer Science and Information Technologies*, pp. 263-265, 2011.
[5] T. Grigoryan and V. Sahakyan, "Dynamic resource manager for clusters", *Proceedings of Conference. Computer Science and Information Technologies*, 2005.
[6] G. Avellino et al., "The first deployment of workload management services on the EU DataGridTestbed: feedbackon design and implementation", *in Proceedings of the 2003 Computing in High Energy and Nuclear Physics Conference (CHEP03)*, La Jolla, Ca, USA, March 2003.
[7] *Data Grid JDL Attributes DataGrid-01-TEN—0142-0_2* .
*http://www.grid.org.tr/servisler/dokumanlar/JDL_Atributes_DataGrid.pdf*
[8] Т. Ху, *Целочисленное программирование и потоки в сетях*, Москва: Мир, 1974.

## Հերթիկազմակերպման մոտեցում գրիդ միջավայրում, սպասման ժամանակի սահմանափակմամբ

Վ. ՍահակյանևՍ. Պետրոսյան

### Ամփոփում

Ռեսուրսների կառավարման և առաջադրանքների պլանավորումը բազմապրոցեսորային հաշվողական Գրիդ միջավայրերում դժվարին խնդիրներից են: Չնայած նախկինում ստացված էական արդյունքներին, դեռևս առկա են խնդիրներ, որոնց լուծումը կհանգեցնի առավել օպտիմալ ռեսուրսների կառավարման: Ավելացնելով սահմանափակումներ

առաջադրանքներին, հնարավոր է ավելի արդյունավետ դարձնել առաջադրանքների հերթի ղեկավարումը և հասնել էֆեկտիվության բարձրացման: Հիմնական սահմանափակումներից մեկն է սպասման ժամանակի սահմանափակումը: Սպասման ժամանակը դա այն ժամանակն է, երբ առաջադրանքը կարող է սպասել մինչև կատարման անցնելը: Սույն հոդվածում դիտարկվում է առաջադրանքների հերթի կառավարման նոր մոտեցում: Այն տալիս է ընդհանրացված լուծման մեխանիզմ, որը կարելի է զարգացնել, հետերոգեն համակարգերի համար ավելացնելով նոր ֆունկցիոնալություն: Հոդվածում առաջարկվում է հերթի կազմակերպման պարամետրացված երկու մոտեցում ՏԻՖՕ վարքով՝ սպասման ժամանակի սահմանափակմամբ: Կատարվում է մոդելների համեմատություն և նշվում են մոտեցումների հիմնական կիրառման պայմանները:

## Организация очереди в Грид среде с ограничением временим ожидания

В. Саакян и С. Петросян

Аннотация

Управление ресурсами и планированием задач в многопроцессорной среде вычислительной системы Грид является одной из труднорешаемых задач. Несмотря на значительные результаты, полученные ранее,остаются проблемы, решение которых приводит к оптимальному управлению ресурсами.Путем добавления ограничений на поставленные задачи возможно увеличить эффективность управления очередью задач и в итоге достичь повышения эффективности.Одним из основных ограничений является ограничение временим ожидания.Время ожидания, это время, в течение которого задача может ждать выполнения.

В данной статье рассматривается новый подход к управлению очередью задач. Он дает обобщенный механизм решения, который можно развивать, добавляя новые функциональные возможности для гетерогенных систем. В статье предлагаются две параметризованные модели управления очередью с ограничением времени и cFIFO поведением. Предоставляется сравнение моделей и отмечаются основные условия применений.