Task Allocation for Computer Service Provider by Optimal CPU Usage Consideration

Somchai Prakancharoen

Department of Computer Science and Information

Faculty of Applied Science King Mongkut's University of Technology North Bangkok

1518 Pracharaj-1 Bangsue Bangkok Thailand

spk@kmutnb.ac.t

***Abstract*—Task allocation algorithm in distributed computing system, such as cloud computing, are now critical issue. Service manager: SM has to consider which Service provider: SP is suitable to service data processing or even data storage from Service request: SR. Normally, SP has to handle its own tasks thus new allocated task should not consume more CPU's work load status which could make SP decrease its base performance. This research objective is to invent the allocation algorithm by study status of SP's CPU usages. The less CPU performance usages: LCU of each SP were specified. The status of each SP's LCU should be used to represent its status which will be a criteria for making a decision by SM whether this SP is suitable to service request from SR or not. The start time and finish time of SR's task was defined by maximized Lagrange transformation function. The experiment is presented in practical calculation. It could point out that some request of SR is more suitable sending to which SP.**

Keywords—Task allocation algorithm; Distributed system; Lagrangetransformation.

I. Introduction

Service manager: SM is a server which responsible in decision making of sending task from Service request: SR, which was initiated from a specific client, to process in suitable Service provider: SP. There are many designed algorithm were used to help SM in making a good decision. A good algorithm should utilized CPU, RAM or even Data storage of SP facilities usage. The allocated task may not tightly disturb or corrupt SP's handled task. The good algorithm ought to support SR in task service processing with rapid response-finishing time.

The objective of this research is to design a simple concept of optimal task allocation from SR to SP under consideration by SM.

II. Related Research And Theory

A. Related Research

[1] First come first served [1] was a technique that each task was allocated to one of SPs that was managed in queue of service. This algorithm is a easiest ones but it may cause serious to SP which are on very busy state and should delay finish time of that task processing. This method is suitable for SP that mostly available to service.

Ant colony [2] is an optimization technique that is applied to task allocation algorithm. Application was split in to many tasks which were considered to distribute to cloud of SP under behavior migration of each SP. This algorithm was designed to prevent work over loading occurring on SP. This technique is very complex method and it was tended to consume plenty of processing time so that it could cause bottle neck at SM.

Load balancing [3] is a most presented algorithm which the algorithm try to study SP's prior knowledge of their loading so that SM should use these information to consider which SP is optimized SP in provide processing for a suitable request task of SR.

The mature techniques for good task allocation algorithms are not yet finished announced. It still be in invention and contribution by many researchers. Thus, this research is also follow the same direction to [2] but reduce its complexity to be a simple one. A simple mathematics technique and dynamic programming are chosen to solve a task allocation problem.

B. CPU usage.

CPU usage in this research was defined that SP and SM have identical CPU performance scale so that calculation of SP's CPU performance usage can include SR's CPU performance usage. In normally, SP's CPU usage has its own likely pattern in a period of time. This is a condition of this research that SP's CPU usage must be similar defined as a specific equation. SR's CPU usage pattern is also performed in the same direction.

C. Lagrange transformation technique[4]

Lagrange transformation is a technique that was used to minimization or maximization a objective function according to constraint functions. If is an objective function is $f(x_1,...,x_n;\beta)$.The constraint function is $g(x_1,...,x_n;\beta)=c$. While x_1 is a random variable, c is a constant and β is vector of parameters. The Lagrange function is $L=f(x_1,...x_n;\beta)+\lambda(c-g(x_1,...,x_n;\beta))$. The solutions to the problem are to derived first order condition; FOC of $c-g(x^*_1,...,x^*_n;\beta)=0$ and $f(x^*_1,...x^*_n;\beta)-\lambda^* g_i(x^*_1,...,x^*_n;\beta)=0$

III. Task Allocation Algorithm Design

Design algorithm of task allocation was begin on prepared relate data, calculate of SP and SR information then make a decision.

A. *Definition of CPU Usage*

In experiment on this research, CPU usage is categorized in three status (Ready: R, Busy: B and Very Busy: VB).The status is defined as shown in table I.

TABLE I. SP Status Definition

SP Status	CPU-performance	RAM Usage
Ready	≤10%	≤4 GB
Busy	10%<and≤30%	≤12GB
Very busy	30%<and≤80%	≤20GB

B. *SP & SR- CPU usage pattern*

For this experiment, CPU performance-usage of sample SP and SR were captured for a period of time as shown in table II and III. This data were curve fitting by mathematic equation , under the best value of R^2, as shown in equation 1 and 2.

$$SP=-0.097*ST^2+4.492*ST+18.497$$

$$R^2=0.911 \text{ while (SP time) } ST\le 50 \quad (1)$$

$$SR=0.032*CT^2-0.873*CT+6.312$$

$$R^2=0.901 \text{ while (SR time) } CT\le 15 \quad (2)$$

Table II. SP CPU usage data

Time	SP : CPU percentage performance
1	12
2	18
3	25
4	32
5	45
6	60
7	50
8	65
9	55
10	50
...	...
47	16
48	11
49	12
50	8

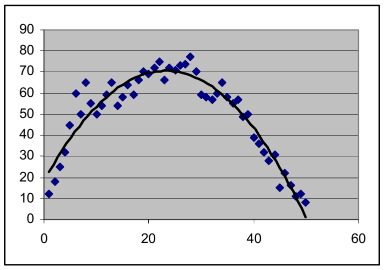
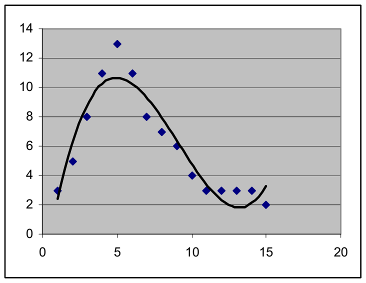


Fig 1. Scatter plot of SP CPU usage

Table III. SP's CPU usage data

Time	SP : CPU percentage performance
1	12
2	18
3	25
4	32
5	45
6	60
7	50
8	65
9	55
10	50
...	...
47	16
48	11
49	12
50	8

Figure 2. Scatter plot of SR's CPU usage



C. *Total CPU usage*

When SR initiate request to SM at time t = n and if SM has to perform calculation for decision making about 3 time unit thus SP should be start it working on t = n + 3. SR has to give more information about its desirable time. For example, if the processing time in SP must be finished before m unit of time, t = n + 3 + m, thus SP has to conduct processing for SR during Ststart = n + 3 and Ststop = n + 3 + m. Each SP was assumed that loading calculation must be considered on this time. The start time of SP, for process SR's task, is at St1(or Ststart) and stop processing at St2(or Ststop). If we assume that each SP has to be consumed its CPU performance in whole time (n+3 and n+3+m) thus for the group of SP, it should be some SP which consume least CPU performance usage. This SP should be a chosen to perform SR task.

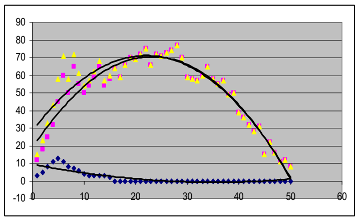


Fig 2. Scatter plot of SP ( ), SR(•) and

Total CPU usage( ).

D. *Practical example*

In figure 2, total CPU usage is calculated from integral summation function of SR and SP during time n+3 to n+3+m. The CPU performance usage during this time, not exceed than 95% (for example) CPU performance usage, is calculated as 95*((n+3+m)-(n+3)) or 95*m. If some SP has large area of idle CPU usage, 95*m minus total CPU usage, then this SP is suitable to be chosen for task allocation. This amount of idle area may be called as loose CPU performance usage: LCU. LCU could be defined as equation (3).

$$LCU = 95*m - \int_{St_1}^{St_2} [(-0.097St^2_1 + 4.492St_1 + 18.497) + (0.032St^2_1 - 0.873St_1 + 6.3120)]dSt \tag{3}$$

or

$$LCU = 95*m - \int_{St_1}^{St_2} (-0.065St^2_1 + 3.619St_1 + 24.809)dSt \tag{4}$$

After numerical of equation (4), the result as shown in equation (5)

$$LCU = 95*m - \int_{St=St1}^{St=St2} [-0.065St + 3.619St + 24.81]dSt \tag{5}$$

While St is started at n+3 and fished at n+3+m. Ct is belonging to St. The LCU value could be calculated by replace constant "n" ,"m"(or St_1= n+3 and St_2=n+3+m).

E. *Objective and constraint function*

SR's task should be start at any time between n+3 to n+3+m. If this task be started at a suitable time, it should give a minimized LCU. That is the objective and constraint functions are shown on equation (6), (7), (8) and (9).

$$Min_{(St_1,St_2)}\{(90*m) - [[(Eq_1atSt_2) + (Eq_2atCt_2)] - [(Eq_1atSt_1) + (Eq_2atCt_1)]]\} \tag{6}$$

or

$$Min_{(St_1,St_2)}\{(90*m) - [[(Eq_1atSt_2) + (Eq_2atSt_2)] - [(Eq_1atSt_1) + (Eq_2atSt_1)]]\} \tag{7}$$

Subject to:

$$n + 3 \le Ct_1 < Ct_2 \le n + 3 + m \tag{8}$$

The Lagrange transformation of (6), (7) and (8) can be present on equation (9).

$$\ell = \{\{(95*m) - [[(Eq_1atCt_2) + (Eq_2atCt_2)] - [(Eq_1atCt_1) + (Eq_2atCt_1)]]\}$$
$$-\lambda_1(n + r - Ct_1) - \lambda_2(Ct_1 - s - Ct_2) - \lambda_3(Ct_2 + w - n - m)\} \tag{9}$$

While r, s and w are filled constant value that use to adapt in equation (5), (6) to be simple equation. After perform derive equation(9), $[\frac{d\ell}{dCt_1}, \frac{d\ell}{dCt_2}, \frac{d\ell}{d\lambda_1}, \frac{d\ell}{d\lambda_2}, \frac{d\ell}{d\lambda_2}, \frac{d\ell}{dr}, \frac{d\ell}{ds}, \frac{d\ell}{dw}]$, then set all results to value "0". The value of Ct_1, Ct_2 are shown as follows.

$$Ct_1 =$$

$$Ct_2 =$$

F. *SM decision making*

After LCU of each SP are calculated then SM choose to allocate SR-task to one of SPs that has the largest LCU.

IV. Summary And Suggestion

The designed simple task allocation algorithm in the research is based on basic mathematics concept. The LCU was calculated from the same start point on both SP and SR.

However, this task allocation algorithm is suitable in ordinary working in most SP which value "m" is not too large deviate from "n". If it is not in this condition, the range of start and stop time is large, then Ct should be start at any point of St. Thus difference equation of SR-CPU performance usage should be considered in order to point out which is the suitable period of time for task allocation. At this point SP's CPU performance usage is surely less consumed.

References

[1] Armstrong P., Cloud scheduler: a resource manager for distributed compute clouds, University of Victoria, Canada, 2010.

[2] Ratan Mishra, Ant colony optimization: A solution of load balancing in Cloud, International journal of Web& semantic technology vol.3 no 2. April, 2012.

[3] Nidhi jain kansal, Existing load balancing techniques in cloud computing: a systematic review, Journal of information systems and communication, vol.3 issue 1, 2012.

[4] David Anderson, Markov chains, University of Wisconsin Madison, 2013.