

Performance Analysis of Multi-level Time Sharing Task Assignment Policies on Cluster-based Systems

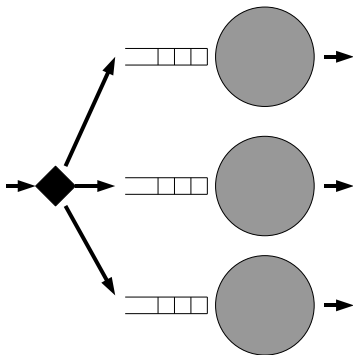
Malith Jayasinghe, Zahir Tari and Panlop Zeepongsekul

RMIT University, Australia

23-09-2010

- Introduction
- Related work
- Problems
- MLMS
- Analysis of MLMS
- Performance Evaluation
- MLMS-M
- Analysis of MLMS-M
- Performance Evaluation
- MLMS-M*
- Analysis of MLMS-M*
- Performance Evaluation
- Conclusion

Distributed System Model



- Cluster-based systems : Cost-effective and scalable.
- Task assignment policy : Assigns tasks to hosts subject to a specific set of rules
- Task assignment policy has a significant impact on the overall performance of the system. Aim is to maximise *performance*

Characteristics of Workloads

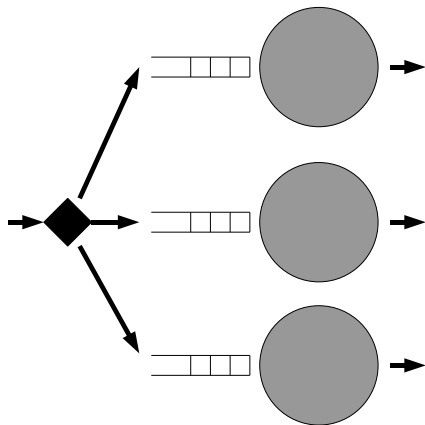
- Computer workloads are highly variable : distributions that represent them are 'Heavy-tailed' [Arlitt and Jin 1999, Arlitt and Williamson 1996, Barford et al.1999]
- Heavy-tailed workloads are represented using Pareto and **Bounded Pareto Distribution**
- Decreasing failure rate
- High variance
- Heavy-tailed distributions have heavier tails than the exponential distribution
 - Unix process CPU requirements ($1 < \alpha < 1.25$)
 - Size of files transferred through the Web ($1.1 < \alpha < 1.3$)
 - Size of files stored in Unix file systems
 - I/O times
 - Sizes of FTP transfers in the Internet $0.9 < \alpha < 1.1$)

- Size-based policies have shown significant performance improvements over traditional task assignment policies under realistic workload conditions (i.e. heavy-tailed workloads).
 - 1. Policies that assume prior some knowledge actual processing requirements (task sizes) of tasks (Type 1)
 - **2. Policies that assume no prior knowledge about actual processing requirements (task sizes) of tasks (Type 2)**

- Our aim is to devise policies to efficiently schedule realistic computer workloads with no prior knowledge about actual processing requirements (task size) of tasks (Type 2)
 - Dynamic web content (PHP, CGI, DB queries)
 - Scientific workloads
- Dynamic content accounts for a significant percentage of web content being requested.
- For this type of tasks, the server is the bottleneck [Cardellini et al., 2002]
- Much existing work assumes processing requirements of tasks are known *a priori* (i.e. static web requests)
- Efficient scheduling of tasks with unknown sizes is a challenging problem : difficult to estimate server load etc.

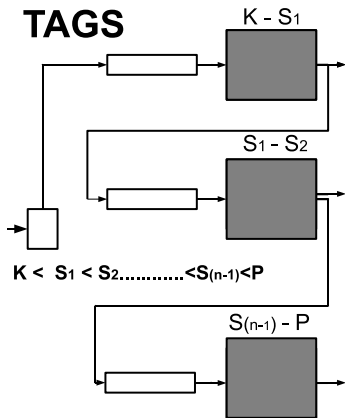
- Random - A traditional task assignment policy
- TAGS (Type 2) : Mor Harchol-Balter (Journal of the ACM (JACM) 2002)
- TAPTF (Type 2) : J. Broberg et al (OPODIS 2004)] and TAPTF-WC (cat 2) : J. Broberg et al (Journal of Parallel Computing 2006)
- SITA-E (Type 1) : Mor Harchol-Balter et al (JPDC 1999)
- LFF-SIZE (Type 1) : Zahir Tari et al (JPDC 2004)
- ADAPTLOAD (Type 1) : Qi Zhang et al (IEEE Transactions on Parallel and Distributed Systems March 2005)

Related Work : Random, Round-Robin



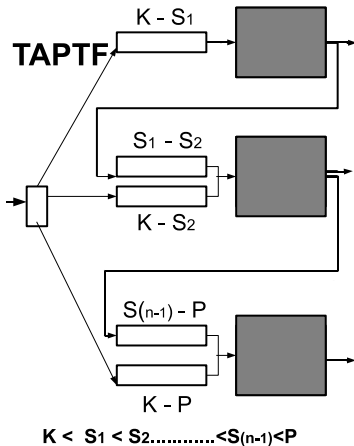
Related Work : TAGS

TAGS (*) [Mor Harchol-Balter (Journal of the ACM (JACM) 2002)]

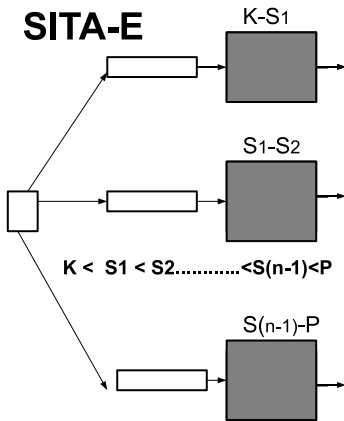


Related Work : TAPTF , TAPTF-WC

TAPTF (*) [J. Broberg et al (OPODIS 2004)] and TAPTF-WC (*)
[J. Broberg et al (Journal of Parallel Computing 2006)]

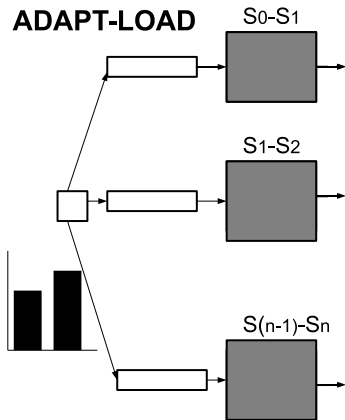


SITA-E [Mor Harchol-Balter et al (JPDC 1999)]



Related Work : ADAPT-LOAD

ADAPT-LOAD [Qi Zhang et al (IEEE Transactions on Parallel and Distributed Systems March 2005)]



Our contribution

- We investigate the performance of task assignment policies in cluster-based systems that support time sharing.
- We focus on heavy-tailed workload distributions and assume no prior knowledge about the actual sizes of tasks.
- We propose 3 models (MLMS, MLMS-M, MLMS-M*), provide an analytical model for each model using queueing theory and investigate the performance (i.e. expected waiting time) of these models under a wide range of workload conditions.

MLTP(Multi-level Time Sharing Model)

- MLTP gives preferential treatment to tasks with short processing requirements.
- MLTP performance well under heavy-tailed workload distributions.
- MLTP does not assume prior knowledge about the actual sizes of tasks.
- How does MLTP work?

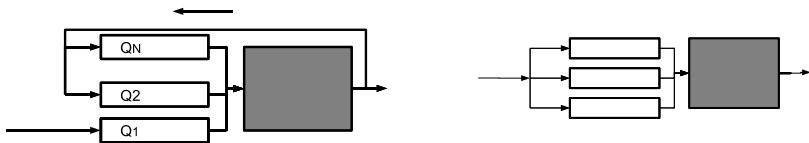
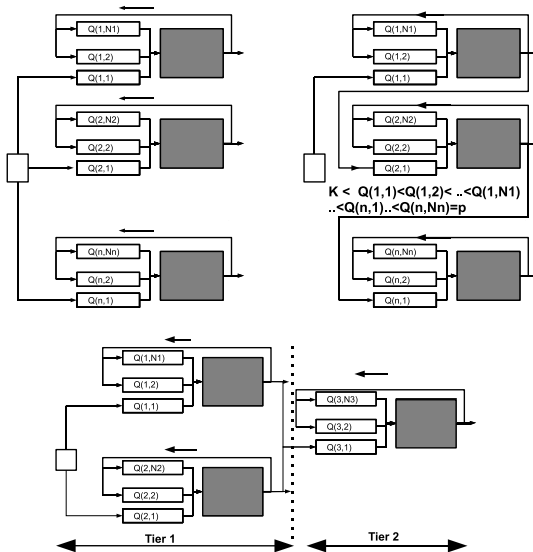


Figure: Priority queue vs multi-level time sharing

Task Assignment Models



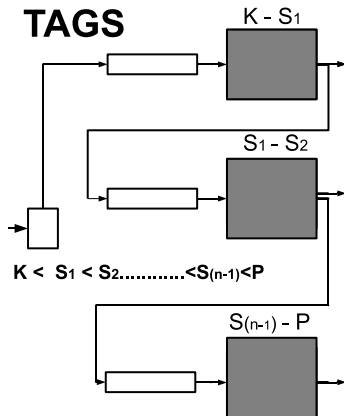


Figure: TAGS

$$f(x) = \frac{\alpha k^\alpha}{1 - (k/p)^\alpha} x^{-\alpha-1} \quad k < x < p \quad (1)$$

- We use Bounded Pareto Distribution to represent heavy-tailed traffic.
- Impact of variability (α) on the performance (X axis = α).
- Performance metric : Expected waiting time (Y axis = $E[W]$).
- System Load : Low (0.3), Moderate (0.5) and High (0.7).

MLMS (Multi-level Multi-server Task Assignment)

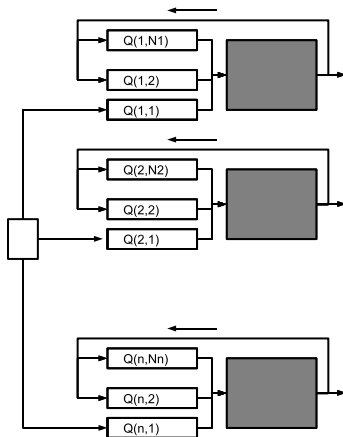
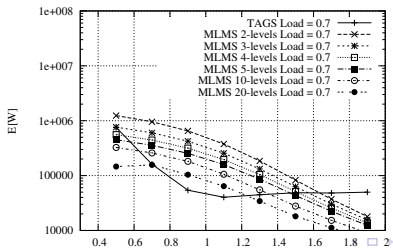
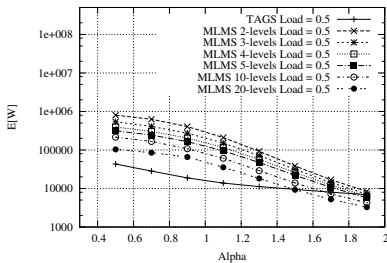
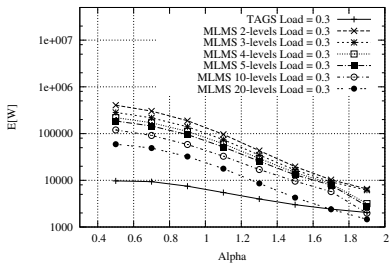


Figure: MLMS

$$E[W]_{MLMS} = \sum_{k=1}^N E[W_k] \int_{Q_{k-1}}^{Q_k} f(x) dx \quad (2)$$

$$E[W_i] = \frac{\lambda E[U_i^2] + \sum_{k=i+1}^N \Lambda_k E[T_k^2]}{2(1 - \lambda E[U_{i-1}])(1 - \lambda E[U_i])} \quad (3)$$
$$+ \frac{Q_{i-1}}{(1 - \lambda E[U_{i-1}])} - Q_{i-1}$$

MLMS : Performance Evaluation



MLMS : Performance Evaluation

- An increase in the number levels results in an improvement in the performance.
- Under the system load of 0.3, TAGS outperforms MLMS for almost all the variabilities (i.e. all α values considered).
- Under moderate (0.5) and high systems loads (0.7), MLMS outperforms TAGS for a range of α values. Under the system load of 0.7, MLMS outperforms TAGS in two different α ranges. For a MLMS policy with 20 levels, these two ranges are 0.4 - 0.7 and 1.3 - 2.0.
- MLMS may require a large number of levels if MLMS is to outperform TAGS. For example, under the system load of 0.7, when $\alpha = 0.5$, MLMS with 20 levels outperforms TAGS by a factor of 5 while under the same conditions, TAGS outperforms MLMS with 2 levels by a factor of 1.5.

MLMS-M (Multi-level Multi-server Task Assignment with Task Migration)

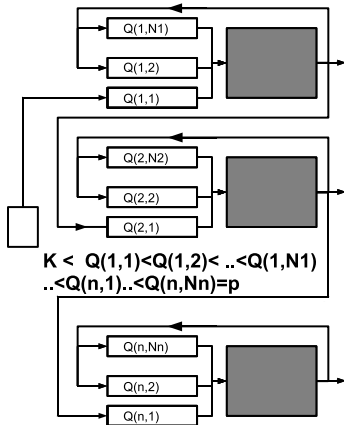


Figure: MLMS-M

Performance Analysis of MLMS-M: Summary

$E[W_{(i,j)}]$ expected waiting time of a task in i^{th} host's j^{th} queue

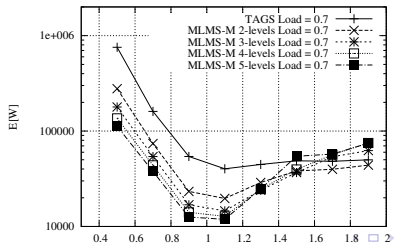
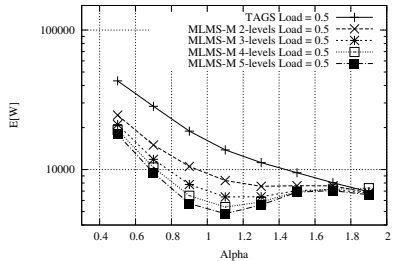
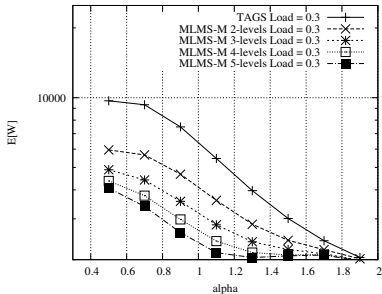
$$P_{(i,j)} = \int_{Q_{(i,j-1)}}^{Q_{(i,j)}} f(x) dx \quad (4)$$

$$P_i = \int_{Q_{(i-1, N_{i-1})}}^{Q_{(i, N_i)}} f(x) dx \quad (5)$$

$$E[W_i] = \sum_{j=1}^{N_i} E[W_{(i,j)}] \frac{P_{(i,j)}}{P_i} \quad (6)$$

$$\begin{aligned} E[W] = & E[W_1]P_1 + (E[W_1] + E[W_2])P_2 \\ & + \dots + (E[W_1] + \dots + E[W_n])P_n \end{aligned} \quad (7)$$

MLMS-M : Performance Evaluation



MLMS-M : Performance Evaluation

- Performance improves with the number of levels.
- MLMS-M outperforms TAGS for almost all the cases considered.
- The highest improvement in the performance is seen under high system loads and very high task sizes variabilities (i.e. low α values).
- For example, when the system load is 0.7 and α is 0.5, MLMS-M with 2 levels outperforms TAGS by a factor of 2.7. Under the same conditions, MLMS-M with 5 levels outperforms TAGS by a factor 6.75.
- Under a fixed system load the expected waiting time does not always decrease with increasing α . This is particularly the case under moderate to high system loads when the number of levels is relatively high.

MLMS-M : Performance Evaluation

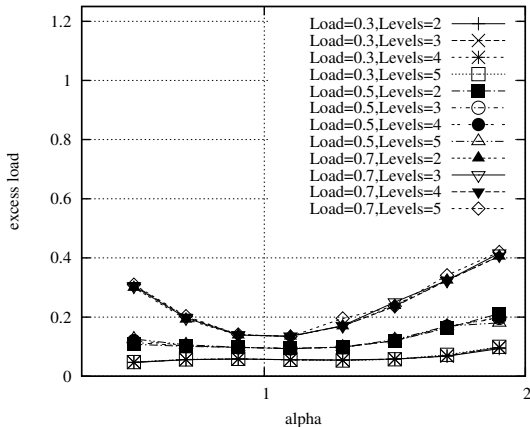


Figure: Excess load under MLMS-M

MLMS-M : Performance Evaluation

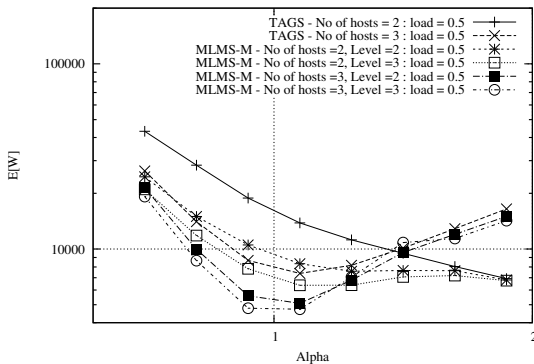


Figure: MLMS-M: The impact of hosts on the expected waiting time

MLMS-M* (Multi-level Multi-server Task Assignment with Task Migration*)

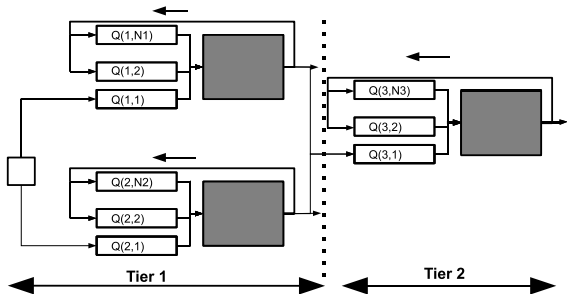


Figure: MLMS-M*

Performance Analysis of MLMS-M*: Summary

$E[W_{(i,j)}]$ expected waiting time of a task in i^{th} host's j^{th} tier

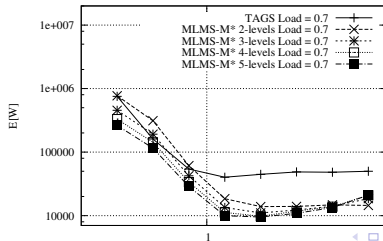
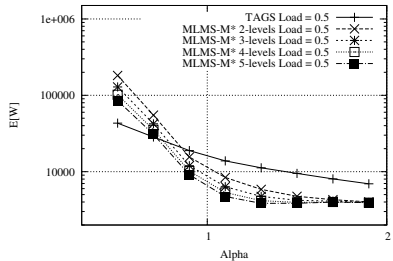
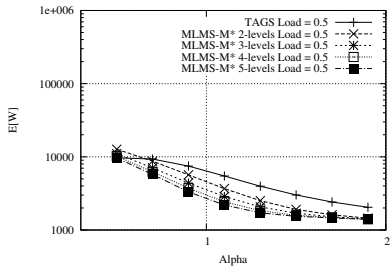
$$P_{(i,j)} = \int_{Q_{(i,j-1)}}^{Q_{(i,j)}} f(x) dx \quad (8)$$

$$P_i = \int_{Q_{(i-1, N_{i-1})}}^{Q_{(i, N_i)}} f(x) dx \quad (9)$$

$$E[W_i] = \sum_{j=1}^{N_i} E[W_{(i,j)}] \frac{P_{(i,j)}}{P_i} \quad (10)$$

$$\begin{aligned} E[W] = & E[W_1]P_1 + (E[W_1] + E[W_2])P_2 \\ & + \dots + (E[W_1] + \dots + E[W_n])P_n \end{aligned} \quad (11)$$

MLMS-M* : Performance Evaluation



MLMS-M* : Performance Evaluation

- MLMS-M* maintains satisfactory performance under high α values.
- Unlike in MLMS-M, there is no significant performance degradations under high α values for all 3 system loads considered.
- MLMS-M* generates relatively low excess.

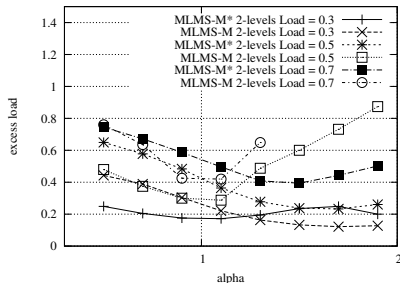


Figure: Comparison of Excess Load : MLMS-M and MLMS-M*

Conclusion

- We provided detailed performance analysis of 3 novel task assignment policies (based on MLTP).
- The analysis assumed heavy-tailed workload distributions because heavy-tailed distributions have been proven to represent many realistic computer workloads.
- MLMS reduce the variability of tasks within hosts and MLMS-M and MLMS-M* reduced the variability of tasks at host level and within hosts and it MLMS outperformed TAGS under specific workload conditions.
- MLMS-M outperformed TAGS for all the scenarios considered. The most significant performance improvement is seen under high task size variabilities and high system loads. Under low task size variabilities MLMS-M generated large amount of excess load resulting its performance to degrade.
- MLMS-M* addressed this issue via its multi-tier host architecture. It outperformed TAGS and MLMS-M under high α values.