Self-Organizing Fair Job Scheduling Among Mobile Devices

Karin Anna Hummel and Harald Meyer
University of Vienna
Department of Distributed and Multimedia Systems
karin.hummel|harald.meyer@univie.ac.at

Abstract

Ubiquitously available mobile devices can contribute to grids not only for accessing resources but also to provide resources, such as, computational power or memory in mobile scenarios. When utilizing networked mobile resources challenges arise due to, e.g., disconnections, disturbances on the wireless medium, and limited energy sources.

We propose an opportunistic job scheduling approach to harness cycles. Mobile nodes decide autonomously which job to take by matching the job's requirements against their capabilities and coordinate among each other based on shared job queues. Reactive and proactive fault tolerance mechanisms assure robustness.

The main contribution of this work is the introduction of five different fairness strategies for self-organizing balanced load distribution based on gossiping about the other mobile nodes' states and comparing the own status against this information. In simulation runs, we achieved an improvement of fair load distribution which remained robust even in the presence of faults.

1 Introduction

Mobile devices are ubiquitously available and allow to build spontaneous mobile ad-hoc distributed systems which can share their processing power and memory. Although connected mobile devices cannot compete with High Performance Computing (HPC) clusters or desktop grids, they still may support harnessing computing cycles.

The envisioned *mobile grids* are expected to be either integrated into stationary grid infrastructures or rather small ad-hoc networked systems useful in mobile field work scenarios which occur in domains like archeology, ecology, or biology, where pre-analysis of data might be desirable. Beyond scientific computing, mobile grids can be used to aim at cooperative energy saving, to achieve a higher degree of connectivity in multihop wireless mesh networks, and sharing of resources like sensors or memory.

In accordance with Satyanarayanan [19], among the most important challenges for mobile distributed computing are lower wireless network bandwidth when compared to wired networks and varying link quality, limited capabilities and energy sources, and varying availability of resources. We, thus, propose a robust scheduling approach targeting these issues. The approach is fully decentralized and fault-tolerant, i.e., tolerating disconnections and low wireless link quality. Job distribution is based on matchmaking between the jobs' requirements and nodes' capabilities. Mobile nodes coordinate the job execution by reading from and writing to shared job queues. In previous work, we described a prototypical implementation which demonstrated the feasibility of the approach based on a distributed Virtual Shared Memory (VSM) [7].

In mobile scenarios, resources are scarce and mobile devices could act selfish, e.g., to save energy. To address such problems, this work extends the basic approach by introducing new strategies to achieve a fair load distribution among the mobile nodes in self-organizing manner. Hereby, load distribution is assumed to be fair, if all nodes execute an equal number of jobs within an observation period. A mobile node is informed about other nodes' performance values by means of gossiping and decides whether it is still willing to process jobs. For local decision making, five different strategies are introduced and investigated in terms of emerging system wide fairness.

The paper is structured as follows: After a survey on related work in Section 2, we describe the decentralized scheduler and the self-organizing fairness approach in Section 3. In Section 4 we present experimental results of a discrete event simulation and conclude our work in Section 5.

2 Related Work

Distributed job scheduling among mobile clusters and grids has recently attracted several research efforts, addressing challenges like energy efficiency [22] and connection failures. Disconnections of wireless links can be masked from applications as proposed for *reliable sockets* in [21] on

network socket layer. From the application's perspective, the disconnection is seen as a kind of suspended network state and connections can be reestablished including the management of possible changes of IP addresses. In [11] probing and monitoring networks to detect wireless channel failures leading to MPI failures are described which allows to invoke fault tolerance mechanisms in time.

Common methods to avoid coordination failures due to resources going offline are using proxies, checkpointing, and replication. In [15] job proxies are proposed to overcome the problem of disconnected mobile devices by maintaining the communication to grid services on behalf of the mobile devices. An example of a distributed checkpoint algorithm which tries to minimize message overhead is given, for instance, in [12]. In [10] task replication is triggered based on the assumption that the failure model of the mobile grid follows a Weibull distribution. In contrast, we use current performance values and estimates about future performance values based on linear regression of recent history data to determine the likelihood of failures and the activation of proactive fault tolerance mechanisms.

In [6] a workqueue with replication is presented, where jobs are replicated among servers without prior knowledge of the worker's condition. On resource failure, replicas are restarted until a given maximum number of replicas is reached. In [1] the concept of workqueues is extended by checkpointing which enables to continue job processing from the last checkpoint. In [20], reliable and unreliable workers are grouped to maximize the probability that each group returns a result. In our work, redundant execution and the use of super-peers are combined to tolerate faults.

In [5] job scheduling is performed via agents that individually decide which tasks to run on which machine. Decisions are based on simple local scheduling rules. Results show, that a tree-based network evolves where important tasks are assigned to more reliable resources which results in improved global network performance. Based on similar design decisions, we introduced super-peers for handling the most reliable operations.

In [2] a framework for opportunistic cluster computing using JavaSpaces is implemented. Similar to our work, this framework uses a shared communication space and autonomous scheduling decisions but does not consider mobility related problems. Based on self-monitoring of utilization, workers decide upon job execution. A distributed load balancing approach is discussed in [4]. Mobile agents monitor the utilization of nodes and encourage the most and the least utilized nodes to balance their workload. In our approach, fair job distribution emerges from the local behavior of the nodes without an additional controlling instance.

The issue to quantify *fairness* in flow control has been for instance addressed in [14]. According to [14] fairness can be thought of as a situation in which no individual class or user is denied access to the network or overly exploited. The Nash equilibrium and competitive equilibrium are points where these two conditions are fulfilled. Especially if all resources are equal, then the throughput at each resource is the same. However, it is also stated that Pareto-optimal points can provide higher user performance. Fairness may be further defined based on different aims, like proportional fairness [8] used to maximize network throughput under the condition that all users may use a minimal amount of throughput. Max-min fairness [3, 16] allocates resources to users in order of increasing demand. In [18] fairness is defined as the standard deviation of dissipated energy between nodes. Values close to zero indicate high fairness. This is similar to our approach, where the variance between the number of processed jobs per node is used as a measurement of fairness.

3 Fair Self-Organizing Job Scheduling

The scheduling approach follows the concept of decentralized opportunistic batch scheduling. As we focus on how to coordinate *unreliable* mobile peers in wireless adhoc grid environments in an efficient and robust manner, we simplified the job model by assuming no inter-job dependencies and equal job priorities. The failures considered, are node failures due to lack of power or disconnections caused by mobility or disturbances on the wireless medium.

The decentralized scheduler is based on the principles of *autonomous local scheduling decisions* and *job queue based coordination*. It is extended by including gossiping about the nodes' states to achieve fair job distribution. Here, fairness is defined as the variance between the total number of processed jobs per node. A possible drawback of this definition is that it does not take nodes with varying resources into account [13].

To avoid a single point of failure, each peer decides autonomously whether to process a submitted job depending on its current and predicted near future capabilities. The coordinated behavior emerges by communicating the status of jobs and storing them in appropriate queues (FIFO order). We chose to base the approach on a distributed Virtual Shared Memory (VSM) because a VSM allows to easily implement the queue based concept and supports asynchronous communication and persistence of data which is beneficial for unreliable mobile nodes. Since the VSM is distributed (using replication of data for Fault Tolerance (FT) and speed up reasons), mutually exclusive consistent data access has to be provided by means of basically locking and reliable replica update mechanisms. Problems occur, if the device holding a lock disconnects.

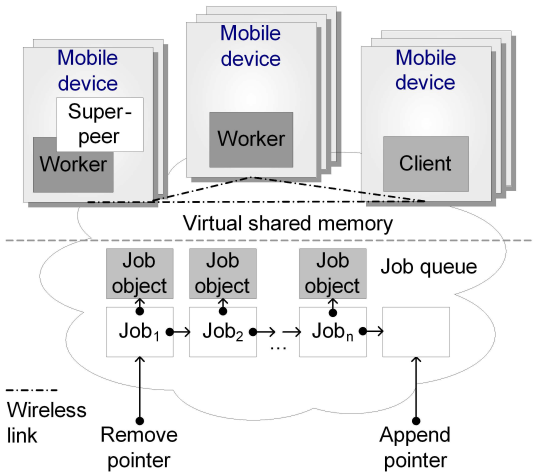

Figure 1. Overview of the system architecture including the job queue concept.

3.1 Self-Organizing Scheduling Approach

The scheduling decisions are taken locally by the mobile peers. The algorithm analyzes the current and predicted future status of the node in terms of connection quality, remaining capacity (battery), utilization, and static resource capabilities. This status is matched against the job's requirements – as given by the job's meta-data – considering also user-configurable policies defining the desired degree of resource contribution. The prediction of future performance values makes use of linear regression.

For the mobile grid scenario, three roles are defined as depicted in Figure 1. A *client* submits jobs to the scheduler. A *worker* performs self-evaluation, matchmaking between job requirements and device capabilities, and job processing. The *super-peer* has been introduced for executing critical tasks such as maintenance, monitoring, and FT services. Each mobile node may host multiple roles simultaneously.

To manage distributed peer collaboration and coordination, queues are used. Basically, jobs are submitted to a queue from where a worker retrieves a job and its corresponding job object which contains job meta-data. Jobs are retrieved in a first come first serve manner. We introduce four queues for job management. Three of them correspond to the job's state (submitted, currently processed, done). A forth queue is introduced which consists of jobs that have to be redundantly processed by another peer since the first worker taking the job considered itself to be unreliable. Peers search for jobs in the queue for submitted jobs (which need to be processed) and in the queue for jobs requiring redundant execution.

The concept includes both reactive and proactive FT mechanisms. Reactive mechanisms try to reestablish a consistent queuing system in the VSM in case failures caused lost locks. Since these mechanisms are costly and most probably lead to job losses in the system, proactive FT mechanisms try to avoid these faults by predicting the node's reliability in the near future.[1]

Node states are introduced corresponding to the likelihood of unreliable operation. The behavior of the mobile peers is determined by these states. To derive the states, the nodes' performance values are evaluated against thresholds for remaining capacity, wireless link conditions, and utilization. Four different node states are defined: *very stable* (a device which remained constantly available in the past and is supposed not to degrade in the future; for super-peers, this node state is required) *stable* (first degradation mode; highly critical tasks are not allowed), *unstable* (the likelihood of disconnecting is high, and therefore proactive FT mechanisms are invoked; nodes are not allowed to hold locks on queue elements and use super-peer services to consistently access and update queue elements; mobile peers redundantly process jobs to increase reliability), and *too unstable* (the node is too unreliable to contribute as a worker). By changing the states, the node's roles might be changed from, e.g., super-peer and worker to being a worker only leading to a decreased number of super-peers in the system. In case no super-peer remains, the system gracefully degrades to working without reactive and proactive FT mechanisms.

We provided a prototypical implementation to demonstrate the feasibility of the basic approach based on the VSM CORSO [9], Java, and the Condor ClassAds [17] in previous work [7]. The speed-up results achieved by a small set of cooperating peers were encouraging, but fairness has not been addressed so far. However, during experiments considering low traffic resulting in empty queues from time to time, this has been identified as a potential problem in potentially exploiting some devices to the benefit of others.

3.2 Introducing Fairness

The basic approach does not assure a fair load distribution between the processing mobile devices although all mobile devices are cooperative. To achieve fairness, we introduce gossiping about node states within a group of nodes and subsequent comparison with the node's own state.

Each mobile node maintains a tree T consisting of n nodes ($n \geq 1$). The root of T is the node T_0 itself, while – if exist – the children $T_1, \cdots T_{n-1}$ are nodes termed *neighbors* (e.g., neighboring nodes in a multi-hop network) from whom node T_0 receives their current performance data to-

[1] Since prediction is not always possible or accurate, proactive mechanisms only complement, but do not replace reactive mechanisms.

gether with a timestamp.[2] The tree-based approach has been chosen to support multi-hop networks as well.

To decide whether to refrain from taking a job, the node evaluates the performance values received according to a strategy. Three types of strategies are considered: First, a lazy strategy type which encourages nodes rather to wait for another node taking a job (*Not best*), second, an assiduous strategy which encourages nodes rather to take a job than to wait (*Worst*), and, third, strategies that evaluate against the average performance or majority in the system (*Worse than average*, *Worse than the majority*, and *Equal or worse than the majority*). In detail, the strategies are described as follows for $n \geq 2$ (where $f(T_i)$ is a function aggregating the monitored performance values, i.e., remaining capacity, utilization, and wireless link condition):

Not best. No job is taken, if

$$\exists\, T_i : f(T_i) > f(T_0), \text{ where } 1 \leq i \leq n-1.$$

Worst. No job is taken, if

$$\forall\, T_i : f(T_i) > f(T_0), \text{ where } 1 \leq i \leq n-1.$$

Worse than average. No job is taken, if

$$\frac{1}{n-1}\sum_{i=1}^{n-1} f(T_i) > f(T_0).$$

Worse than the majority. No job is taken, if

$$|\,\{T_i \;:\; f(T_i) > f(T_0), 1 \leq i \leq n-1\}\,| > \frac{n-1}{2}.$$

Equal or worse than the majority. No job is taken, if

$$|\,\{T_i \;:\; f(T_i) \geq f(T_0), 1 \leq i \leq n-1\}\,| > \frac{n-1}{2}.$$

Additionally, a deadlock prevention mechanism has been included which deactivates each strategy and degrades to basic job management without fairness in case jobs remain in the queue for submitted jobs too long (this timeout is configurable).

The size n of the non-disjoint group can be configured. The group forming algorithm used should assure that the network is not split which would prevent network wide self-organization. In the current implementation, we assume a unique node ID, like the MAC address, known by the other nodes in range of the ad-hoc network. After sorting (the highest node ID should be the predecessor of the lowest one to build a ring), each node selects the $\lfloor (n-1)/2 \rfloor$ nodes preceding its ID and the $\lceil (n-1)/2 \rceil$ nodes following to create its neighbor group. (In case n equals the number of nodes in the network, every node will send its performance data to each other node.) Assuming non-cheating mobile peers, fair job distribution should emerge.

[2]For reasons of simplicity, we assume synchronized clocks.

3.3 Simulation Framework

For the evaluation of the fairness strategies a simulation framework based on Discrete Event Simulation (DES) was designed based on the insights derived from a prototypical implementation of the decentralized scheduler [7]. The simulation approach allowed us to vary important parameters for evaluating fairness. The simulator implements a model of the VSM, but abstracts from lower wireless network layers while only considering network parameters important for the scheduling approach, like the signal strength of the wireless link.

The simulation framework is entirely object oriented and plugin-based which allows to extend the framework easily. All parameters can be controlled via a GUI and by a timeline which is defined in a separate XML file. The entities of the simulations are mobile nodes (i.e., their properties, such as, FLOPs and processor utilization), wireless links (modeled based on the free space loss properties), an energy source model (linear capacity loss over time), and a movement model (Random Waypoint or timeline based).

4 Experiments

Experiments were conducted using the DES presented in Section 3.3 to investigate the fairness strategies in faultless and faulty scenarios. The correctness of the simulation model was validated against a prototype implementation [7], which already demonstrated the feasibility of the decentralized scheduling approach. Although the simulation model exhibits some simplifications, it shows the same trends in average response time as the prototype and varies from the prototype in the same range as different prototype runs vary.

The job processing time is assumed to be 116.632 seconds (as has been measured for an example test RC5 key decoding application for calculating a 24 bit key on a worker notebook). The simulated mobile scenarios consist of two nodes hosting two super-peers and one client, while the number of workers is $N = 20$. During the experiment runs, the client submitted 60 jobs with a frequency of one job per 110 seconds. This submission frequency was chosen to investigate in particular scenarios where the unfairness of the basic approach becomes evident because only a subset of the nodes can bear the load (in our case, two nodes). The deadlock prevention was configured to deactivate the fairness strategy after ten seconds multiplied by the number of potential candidates for job taking. For investigating faulty situations, we used the timeline based approach for determining disconnected times.

4.1 Fairness Strategies in Faultless Scenarios

Fairness is here defined as the balanced workload between mobile workers and measured in terms of variance of the number of jobs processed by the workers during the simulation time. A low variance indicates a fair strategy.

Figure 2 shows the results for the five fairness strategies compared to the basic *no fairness* strategy with a varying group size n. Larger n lead to better knowledge of performance values of other nodes causing more fairness for most strategies.

Without fairness (*no fairness*) the variance was highest. Among the fairness strategies the strategy *worst* performed worst because it allowed more nodes to participate freely (similar to the *no fairness* strategy) which led to only a few nodes doing all the work. The strategy *not best* outperformed the others in particular for small group sizes. This strategy has been, however, vulnerable to deadlock situations and often invoked deadlock prevention, i.e., system degradation to *no fairness* and changing back to the *not best* strategy, resulting in an increased average job response time.

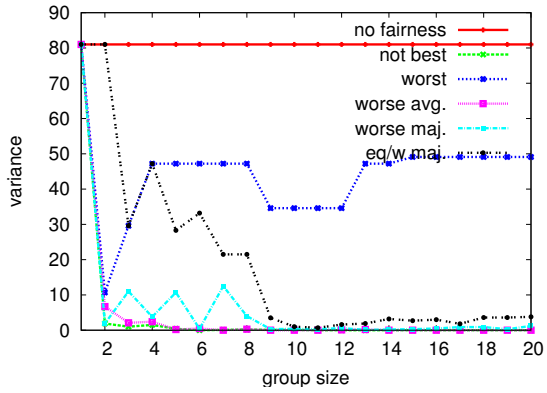

Figure 2. Variance of different fairness strategies with varying group size.

With the group size the number of messages for gossiping the nodes' states increased for each strategy. In principle, the number of messages per communication round is given as $M = n \times N$ which is reduced in the implementation by omitting messages that will not provide new performance data. For example, the measured number of messages is 6680 for $n = 2$, 49432 for $n = 10$, and 102872 for $n = 20$ given a simulation period of 8000 seconds with a round interval of 30 seconds.

4.2 Fairness Strategies in Faulty Scenarios

To derive first results of the fairness strategies in the presence of faults, we set the group size to a fixed value $n = 10$ and assumed one faulty node. A timeline for generating reproducible faulty behavior was used: The node went offline after 100 seconds, recovered after 900 seconds and went offline after 3080 seconds again (times were chosen in such a way that errors are provoked).

Table 1 summarizes the robust behavior of the five strategies and *no fairness*. The faultless case *no error* was added for comparison reasons. When *no FT* mechanisms are invoked, the strategies creating high variance in the no error case (*no fairness*[3], *worst*) benefit from the faults since overloaded nodes may go offline and foster others to take over. The other strategies' variance did not change or increased. When *reactive FT* mechanisms only are invoked, the behavior is similar to the *no FT* case. When *proactive FT* mechanisms were used, most strategies' variance increased. This is due to redundant execution of jobs (and thus, more jobs being executed in the system).

	no error	**no FT**	**proactive FT**	**reactive FT**
no fair.	85.26	17.64	203.67	61.57
not best	0	0.11	0.63	0.11
worst	36.42	28.42	91.95	38.11
worse avg.	0	0.11	0.11	0.11
worse maj.	0.32	0.32	0.99	0.32
eq/w maj.	0.73	3.16	2.85	3.16

Table 1. Variance of different fairness strategies in the presence of faults compared to the faultless case (no error).

5 Conclusions

We proposed a fault-tolerant self-organizing job scheduling approach for mobile ad-hoc grids providing fair load distribution among nodes. Based on self-evaluation of nodes' performance measures and gossiping about nodes' performance values among groups of neighbors, nodes decide whether to take a job or not. We introduced five fairness strategies to support decision making and evaluated them in terms of variance of number of jobs executed by the nodes under varying group sizes.

Each fairness strategy improved job distribution when applied to a system configuration where unfairness is observed without using these strategies. A strategy causing a node to take a job only if the node exhibits the best performance values within the group (together with a graceful degradation of the system to overcome deadlocks) led to the

[3]Note, that due to unrecoverable failures, not all jobs could be finished in this case.

best fairness results but caused additional delays. Fairness strategies evaluation against the majority or average performance values of the group led to good results for higher group sizes. In future work, we plan to detail the trade-off between group size and achievable fairness and to include heterogeneous mobile devices into our model and investigations.

Acknowledgment

The work described in this paper is partially supported by the Austrian Grid Project, funded under contract GZ BMWF-10.220/0002-II/10/2007.

References

[1] C. Anglano, J. Brevik, M. Canonico, D. Nurmi, and R. Wolski. Fault-Aware Scheduling for Bag-of-Tasks Applications on Desktop Grids. In *7th International Conference on Grid Computing*, pages 56–63, 2006.

[2] J. Batheja and M. Parashar. A Framework for Opportunistic Cluster Computing Using JavaSpaces. In *9th International Conference on High-Performance Computing and Networking*, pages 647–656, 2001.

[3] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, Inc., 1987.

[4] J. Cao. Self-Organizing Agents for Grid Load Balancing. In *5th IEEE/ACM International Workshop on Grid Computing*, pages 388–395, 2004.

[5] A. Chakravarti, G. Baumgartner, and M. Lauria. The Organic Grid: Self-Organizing Computation on a Peer-to-Peer Network. In *International Conference on Autonomic Computing*, pages 96–103, 2004.

[6] D. P. da Silva, W. Cirne, and F. V. Brasileiro. Trading Cycles for Information: Using Replication to Schedule Bag-of-Tasks Applications on Computational Grids. In *9th International Euro-Par Conference*, pages 169–180, 2003.

[7] K. Hummel and G. Jelleschitz. A Robust Decentralized Job Scheduling Approach for Mobile Peers in Ad-hoc Grids. In *7th IEEE International Symposium on Cluster Computing and the Grid*, pages 461–470, 2007.

[8] F. Kelly. Charging and Rate Control for Elastic Traffic. *European Transactions on Telecommunications*, 8:33–37, 1997.

[9] E. Kühn. Coordination System. Patent EP0929864 B1, European Patent Office, D–80298 Munich, 2001.

[10] A. Litke, D. Skoutas, K. Tserpes, and T. Varvarigou. Efficient Task Replication and Management for Adaptive Fault Tolerance in Mobile Grid Environments. *Future Generation Computer Systems*, 2007(23):163–178, 2007.

[11] E. Macias, A. Suarez, and V. Sunderam. Efficient Monitoring to Detect Wireless Channel Failures for MPI Programs. In *12th Euromicro Conference on Parallel, Distributed and Network-Based Processing*, pages 374–381, 2004.

[12] Y. Manabe. A Distributed Consistent Global Checkpoint Algorithm for Distributed Mobile Systems. In *8th International Conference on Parallel and Distributed Systems*, pages 125–132, 2001.

[13] G. F. Marias, P. Georgiadis, D. Flitzanis, and K. Mandalas. Cooperation Enforcement Schemes for MANETs: A Survey: Research Articles. *Wireless Communications and Mobile Computing*, 6(3):319–332, 2006.

[14] R. Mazumdar, L. Mason, and C. Douligeris. Fairness in Network Optimal Flow Control: Optimality of Product Forms. *IEEE Transactions on Communication*, 39(5):775–782, May 1991.

[15] S.-M. Park, Y.-B. Ko, and J.-H. Kim. Disconnected Operation Service in Mobile Grid Computing. In *Service-Oriented Computing - ICSOC 2003*, pages 499–513, 2003.

[16] B. Radunović and J.-Y. L. Boudec. A Unified Framework for Max-Min and Min-Max Fairness with Applications. *IEEE/ACM Transactions on Networking*, 15(5):1073–1083, 2007.

[17] R. Raman, M. Livny, and M. Solomon. Matchmaking: Distributed Resource Management for High Throughput Computing. In *7th International Symposium on High Performance Distributed Computing*, pages 140–146, 1998.

[18] A. Safwat, H. Hassanein, and H. Mouftah. Energy-aware Routing in MANETs: Analysis and Enhancements. In *5th ACM International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems*, pages 46–53, 2002.

[19] M. Satyanarayanan. Fundamental Challenges in Mobile Computing. In *15th Annual ACM Symposium on Principles of Distributed Computing*, pages 1–7, 1996.

[20] J. Sonnek, M. Nathan, A. Chandra, and J. Weissman. Reputation-Based Scheduling on Unreliable Distributed Infrastructures. In *26th IEEE International Conference on Distributed Computing Systems*, pages 30–, 2006.

[21] V. Zandy and B. Miller. Reliable Network Connections. In *8th Annual International Conference on Mobile Computing and Networking*, pages 95–106, 2002.

[22] Z. Zong, M. Nijm, A. Manzanares, and X. Qin. Energy Efficient Scheduling for Parallel Applications on Mobile Clusters. *Cluster Computing*, 2008(11):91–113, 2008.