

A Monitoring Architecture for Control Grids

Alexandru Iosup¹, Nicolae Țăpuș¹, and Stéphane Vialle²

¹ Politehnica University of Bucharest,
Splaiul Independentei 313, Bucharest, Romania
AIosup@cs.pub.ro, NTapus@cs.pub.ro,*

² Supélec,
2 Rue Edouard Belin, 57070 Metz, France
Stephane.Vialle@metz.supelec.fr **

Abstract. Monitoring systems are nowadays ubiquitous in complex environments, such as Grids. Their use is fundamental for performance evaluation, problem spotting, advanced debugging and per-use accounting. Building such systems raises challenging issues, like data gathering from Grid components, low intrusiveness, ease of use, adaptive data visualization, fault-tolerance and self-maintenance. This paper presents a new layered architecture, named Toytle, specifically designed to address these issues in the context of control Grids. All their components, from computing and network resources to complete physical processes with soft time constraints, can be monitored with Toytle. The architecture's layers, namely the distributed core, the hierarchical connections and the local monitors, have been designed to ensure scalability, high-speed sampling and efficient dealing with large data bursts. The future Toytle implementation will adapt existing tools and also create entirely new modules.

Keywords: control Grids, monitoring architecture.

1 Introduction

From the traditional computational and data Grids to the recent peer-to-peer and semantic Grids [2], there are many types of Grids and Grid applications that one can see nowadays. One of the most interesting examples is the emerging of control Grids, a special case of service grids [8], offering users the ability to control remote resources, including applications, computers and non-standard physical devices, in standardized ways. In general, control Grids serve as infrastructure for interdisciplinary projects, and are often used for applications adapted from the cluster and multicluster environments [16, 6]. As possible scenarios we mention mobile robotics applications [12] and intrusion detection sensors management, from the physical processes having soft time constraints field, and low-cost local collaborative environments [3], from government/academia.

One of the perennial activities for deploying and maintaining complex environments is monitoring. In particular, monitoring control Grids is a demanding task

* This work was partially supported by the RoGrid national programme [15].

** This work was partially supported by the ACI-GRID programme.

[10]. First, the nature of the resources can be truly heterogeneous, from computing and storing resources to mobile devices and even mobile robots. Second, given the applications often encountered in control Grids, the locality of available resources ranges from constrained (e.g. clusters) to mid-range (e.g. inter-country). Finally, all these resources must be put together in concerted applications, which must be monitored in full. Techniques for monitoring large cluster environments already exist [9] and methods for monitoring computational Grids have already been developed [19]. Yet, it has not been established whether any of these two, or a combination, can be used in the case of control Grids, especially with physical processes having soft time constraints.

This work addresses the issues related to monitoring control Grids, with a threefold contribution. First, it identifies the requirements of control Grids monitoring (see Sect.2). Second, it introduces an architecture that focuses on the identified requirements (see Sect.3). Third, it emphasizes a number of research issues related to the architecture and concerning distributed monitoring data aggregation, storage and retrieval (see Sect.4). We also put our results in the perspective of related work (see Sect.5) and present our conclusions (see Sect.6).

2 Monitoring control Grids

Monitoring in Grid environments typically results in dealing with *Grid-awareness*, *scalability* and *standards-based communication* [4]. In addition, ensuring *low intrusiveness* on the monitored machines, *presenting relevant data* in meaningful ways, and guaranteeing degrees of *fault-tolerance*, should also be enforced.

Monitoring control Grids raises additional problems, like:

- gathering data from all Grid components (network, hosts and devices, middleware and applications),
- dealing with large "bursts" of information, especially for sensors fields, where data acquisition is performed simultaneously and periodically,
- performing high-speed sampling, as non-standard devices must be monitored in full, with many parameters being measured frequently.

Even more, as control Grids are, in general, interdisciplinary projects, with most of the people involved having low technical skills, if some at all, the installation and management of monitoring tools are extremely important.

3 A layered architecture for monitoring control Grids

We have designed a new monitoring architecture, named Toytle, which addresses the main issues of monitoring control Grids and their respective applications (e.g. physical processes with soft time constraints). This architecture follows the Global Grid Forum Grid Monitoring Architecture guidelines (short, GGF GMA [17]), regarding the structure and protocols required for Grid performance monitoring.

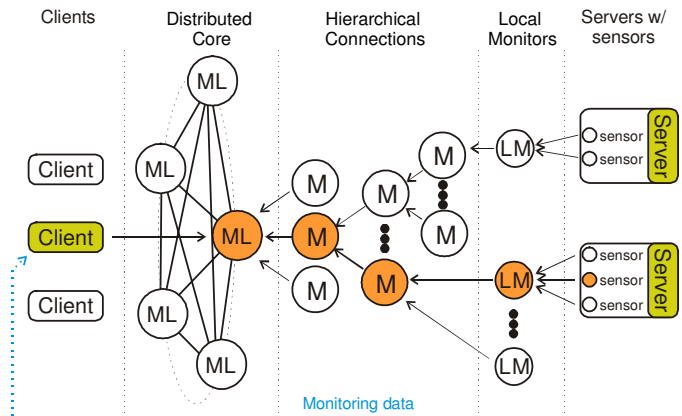


Fig. 1. The Toytle monitoring architecture components: *distributed core*, *hierarchical connections* and *local monitors*.

3.1 Overview

The Toytle architecture is structured in three layers (see fig. 1):

1. **Distributed core layer** - a complex core with near cross-bar connections between nodes that provides high-level monitoring services to the user;
2. **Hierarchical connections layer** - a fairly light hierarchical structure that connects the local monitors to the distributed core, through data federation and piping;
3. **Local monitors layer** - a very light and lowly-intrusive set of monitors (i.e. not just sensors) that gather data from locally available sensors and perform some simple filtering on it.

The three layers cooperate for seamless monitoring services, in a hybrid, near *cross-bar core/N-level hierarchical connections* structure. By *near cross-bar* we mean that nodes inside the core have links to each other, resembling to some extent to a full cross-bar interconnection. By *N-level hierarchical connections* we mean the nodes that are not part of the core form a tree-like structure, with as many levels as needed.

This approach ensures that, while the hierarchical connections lead to a highly scalable approach, the data obtained from multiple large geographical areas (thus, multiple hierarchical trees) can be gathered in a distributed core, with ease of access added together with some degree of fault tolerance.

3.2 The distributed core layer

The distributed monitoring core deals with issues like data gathering, storage, replication and visualization. Having a horizontal structure (see fig.2), with nodes distributed according to geographical or problem-wise needs, this module tackles

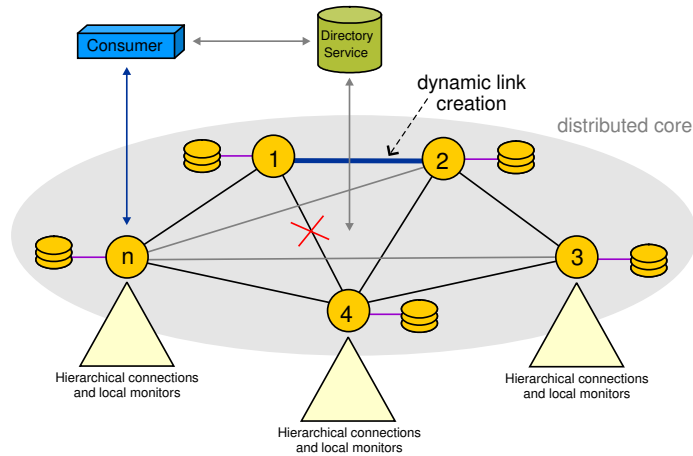


Fig. 2. The Toytle architecture distributed core layer. The nodes within the core can maintain permanent contact (thick connection lines between nodes) or just be able to communicate with one another (thin connection lines between nodes).

issues like adaptive system balancing, fault-tolerance, self-maintenance, Grid data access and user control.

The monitoring core nodes are able to *transparently balance* the monitoring system use. First, when a user tries to connect to a core node, given that the contacted node is too crowded, it will be automatically relocated to another core node. Second, data summarization should be performed by the least charged core node that has access to that data. For both these tasks, the core should also be self-monitoring.

Fault-tolerance is ensured through data replication. Data storage is performed per-node or per-system in one or more repositories. Recent data is stored in cache-like memory buffers, for faster access. Recent data replication is performed transparently for the user between core nodes. Data visualization allows for both resource- and application- oriented views. The monitoring core is self-maintained, so that node insertion and failure are dealt with transparently to the user. A link failure between two core nodes (see the cross in fig.2) can be transparently replaced by another link creation (see the link between nodes 1 and 2 in fig.2).

Also, in the event that the core node that was maintaining the connection with a lower-layer node (a root monitor from the hierarchical connections layer) falls, the core should transparently relocate the now broken connection to another core node. In other words, as long as the core contains at least one node, data is available to the users, and as long as several nodes are active at the core level, data federation is optimized for speed issues.

User control of the monitoring system is done through the specification of monitoring levels for all four categories of Grid monitoring data: host, network,

middleware and user applications. Every characteristic monitored by the monitoring system has a type and a monitoring level associated with it. Once the monitoring level for the characteristic's type has been surpassed, that characteristic becomes monitored and its data starts reaching the core, thus being available to the user. The higher the monitoring level is, the larger the size of data gathered and summarized at the core level gets.

Since the number of core nodes typically does not exceed 1 – 2% of the total number of resources in the monitored Grids, the near cross-bar interconnection at this level does not raise *scalability* issues. The amount of extra data space required for replication depends on the replication strategy (see Sect. 4, *data consistency and data replication* paragraph) and on the filtering options employed by the user. The low number of core nodes makes this extra-resource usage affordable. The fact that the core nodes are not light does not interfere with the *low intrusiveness* requirement, as the core nodes do not actually run on the monitored system's machines.

Finally, in order to comply with the GGF GMA (e.g. consumer / producer / directory architecture), information about the monitoring core's data type and actual data location can be obtained from a monitoring directory. Also, standard data representations, both with size reduction (e.g. XDR [5]) and scheme self-definition (e.g. XML) targets, are offered.

3.3 The hierarchical connections

The *hierarchical connections layer* of the monitoring system performs data gathering and summarizing. It is also responsible for ensuring hierarchical monitoring connections, while keeping the monitored system's load at a minimum, and for transmitting changes in monitoring levels to the local monitors.

The vertical connection structure (see fig.3) allows for efficient data-summarization, aiming to reduce the load of the monitoring system. Each monitoring hierarchy top node is linked with one or more monitoring core nodes. As long as at least one of its nodes is active, the hierarchical connections layer continues to perform as a link between the core and the local monitors. This fault tolerance target is designed to be performed transparently to the user.

Temporary data recovery, in the context of low intrusiveness, is offered through backups on fixed-size (e.g. using a round-robin storage strategy) databases. As for the core layer, standard data representations, both with size reduction and scheme self-definition targets, are offered at this layer.

3.4 The local monitors

The *local monitors layer* performs all the data acquisition and simple on-the-fly filtering. At this level, locally-close monitors are connected in horizontal structures (see fig.4), with important benefits like node insertion and failure being

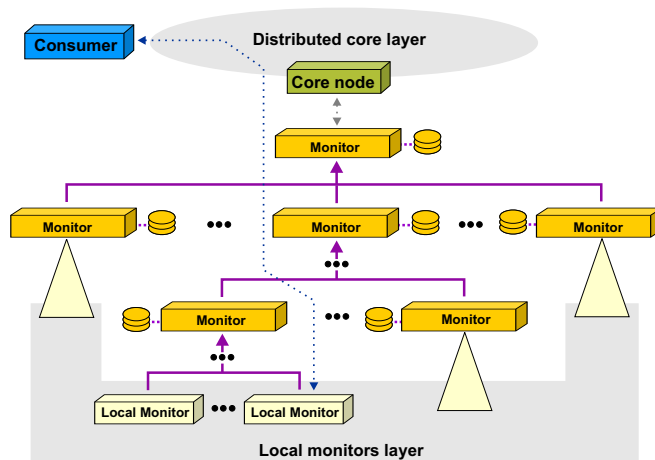


Fig. 3. The Toytle architecture hierarchical connections layer. Monitors form an N-level hierarchical structure. Monitors can store data in local databases. Note that consumers cannot directly access nodes at this level.

dealt with transparently to the user. Data from all of the Grid components, network, hosts, middleware and applications, can be acquired at this level.

Fault-tolerance is ensured through data advertisements. Each local monitor advertises its data and sends it to other local monitors, so that exact replicas of that data can be temporarily found on other places than the origin, at any time.

Simple filtering should be performed at this level. By *simple* we mean filtering characteristics with thresholds and data correlation. Filtering with thresholds is done by setting certain limits, called *thresholds*, such that if a characteristic goes over the threshold, it gets immediately reported. Filtering by data correlation means not only defining thresholds for any given characteristic, but also combinations of thresholds for combinations of characteristics. For example, the condition could be that CPU load is less than 50% *and* available RAM is under 10% of the total RAM. Filtering by any of these means should be done on all Grid data targets: network, hosts, middleware and especially user applications.

Two different *data pruning* methods are used at this level: *derived data pruning* and *monitoring level pruning*. Derived data pruning means that data that can be derived from other available data is not transmitted through the hierarchical connections layer. Pruning according to a specified monitoring level means that each characteristic has a verbose level and must be acquired once the monitoring level reaches it. In any case, the low-intrusiveness requirement is that generated traffic for data transfers should be less than 1% of the theoretical bandwidth of the link (e.g. 0.1 *Mbps* for a 10 *Mbps* Ethernet connection, 1 *Mbps* for a 100 *Mbps* Fast Ethernet connection, and 10 *Mbps* for a 1000 *Mbps* Gigabit Ethernet connection).

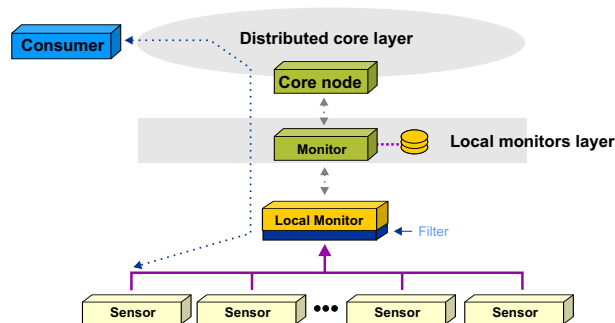


Fig. 4. The Toytle architecture local monitors layer. Local monitors and sensors form a 1-level hierarchical structure. Note that consumers cannot directly access nodes at this level.

The sampling rate of this level must be of at least $10 \text{ samples/second}$. This targeted sample rate has been derived from the normal local area network round-trip time (the time needed for a single packet to get from a source to a destination and back), which is nowadays just below $0.1s$. A recommended sample rate should be, however, around $20 \text{ samples/second}$.

This layer also sends data to the upper layer in standard data formats, like XDR and XML. We emphasize that a compact and portable data transfer standard, like XDR, must be used at this level, because of the sheer size of the monitoring data.

3.5 Towards implementing the monitoring architecture

Our first attempts of implementing the monitoring architecture focused on finding existing open-source monitoring tools, which may be minimally modified in order to integrate within Toytle. This approach allows us to focus on the novel aspects of the Toytle architecture, instead of dealing with already solved issues.

SNMP v2(c)/3 [13] is a flat structure standard for monitoring, which can deal with a large number of network device types, thus being an obvious candidate for the local monitors layer. Unfortunately, our tests on NET-SNMP, a renowned SNMP package, indicate that both its response latency and its generated traffic are too high for our mission.

Ganglia [9] has a hierarchical, tree-like, structure and is well-suited for high-speed data aggregation. The mechanisms employed by Ganglia for fault-tolerance, namely auto-managed multicast channel and periodic *heart-beat* signal, cover very well our requirements in these areas, at the local monitors level. Our tests show that Ganglia should be improved with adequate user data definition, a compressed data exchange protocol within the tree, fault-tolerance, and enhanced data filtering. Therefore, Ganglia is a very good candidate for future integration at the hierarchical connections and local monitors layers.

4 Research issues

In this section we identify a number of open research issues concerning the distributed monitoring data aggregation, storage and retrieval, all in the context of the Toytle monitoring architecture.

Core connectivity factor impact At the extremes, the core can have either ring or full-mesh topologies. In-between, peer-to-peer connectivity and data lookup strategies may be applied [14], to ensure certain performance guarantees. Therefore, the performance of the system with various core topologies remains to be established.

Data consistency and data replication In order to ensure geographical and administrative scalability, several data consistency models can be used for the core layer. Beside the traditional strict consistency, the *conit*-based continuous consistency model [21] seems to be a promising alternative. Further issues regarding replica granularity, replica placement, and user characteristics must be studied for the devise of an adequate data replication strategy.

Data query language The information stored by the monitoring service may be retrieved using targeted queries. It is debatable whether the large amounts of data, with *bursty* behavior, should be accessible through a standard language, like SQL, or through a customized one.

Adaptive caching The low/middle layers of the proposed monitoring architecture deal with repeated data transfers. Is storing data within a fixed-size (round-robin) table enough, or should there be an option for dynamically-sized disk (scratch) tables? Should there be a memory caching before writing to disk? We believe that such questions must find their answers in the context of real-world control Grids applications.

5 Related work

There has been extensive work within the field of monitoring parallel and distributed systems. Recently, much interest has been put into monitoring Grid systems. A good survey on Grid monitoring can be found in [4]. However, we claim that nowadays monitoring systems cannot address in full monitoring control Grids and applications running in such environments.

Two monitoring systems are very close to our view on the monitoring core. First, Astrolabe [18] addresses peer-to-peer distributed information management, with special focus on scalability, but uses full-scale *gossiping* (everyone gets to know everything), which may render the system impractical for real-time data aggregation. Second, MonALISA [11] has a flat structure and is well suited for data visualization and high-speed sampling, but lacks core connections and is not open-source, both of which issues are vital for our purposes.

Systems like DataGrid's GRM/Prove [7] and NetLogger³ lack a lowly intrusive multi-level monitoring hierarchy. The NWS [20] system cannot be used with "bursty" applications and is not suitable for fast changing data sets. Systems like R-GMA⁴ and MDS⁵ are too resource-demanding and/or slow-paced to be effective for the case of control Grids. CrossGrid's OCM-G [1] leaves up to the user to collect the data and share it between the nodes that request it, thus being impractical for control Grids. The GrADS Autopilot [19] toolkit is targeted at automatic performance monitoring and applications steering in the context of computational Grids, thus having a different focus than ours.

6 Conclusions and future work

This paper has presented a layered monitoring architecture, named Toytle, specifically designed for addressing the special requirements of control Grids. The architecture is a hybrid near cross-bar core/N-level hierarchical connections monitoring architecture, that aims at satisfying the most important requirements of control Grids. High speed data gathering and federation, scalability, program lightness (in terms of monitored system's resource consumption), low intrusiveness (in terms of monitored system's performance degradation) and easy deployment are targeted by this architecture. The architecture's three layers, namely the distributed core, the hierarchical connections and the local monitors, ensure a good degree of modularity, while still offering the advantages of architectural compactness.

For the future, we plan to develop new modules for the core layer, as well as adapting Ganglia to the hierarchical connections and the local monitors layers. This will lead to a complete implementation of Toytle, and will give us a good platform for addressing the research issues presented in Sect.4. Also, testing the complete implementation of Toytle against an existing Grid mobile robotics application will help identifying possible improvements to this monitoring architecture.

References

- [1] B. Balis, M. Bubak, W. Furnika, T. Szepieniec, R. Wissmueller, and M. Radecki. Monitoring Grid applications with Grid-enabled OMIS monitor. In *Proceedings of the First European Grids Conference, AxGrid 2003, Santiago de Compostela, Spain*, pages 230–239. Springer Verlag, February 2003.
- [2] F. Berman, A. Hey, and G. Fox. *Grid Computing: Making The Global Infrastructure a Reality*. Wiley Publishing House, 2003. ISBN: 0-470-85319-0.
- [3] G. Fox. Experience with distance education 1998-2003. Collection of resc., <http://grids.ucs.indiana.edu/ptliupages/publications/disted/>.

³ NetLogger, <http://www.didc.lbl.gov/NetLogger>

⁴ R-GMA, <http://www.r-gma.org>

⁵ Monitoring and Discovery Service, <http://www.globus.org/mds/>

- [4] M. Gerndt, R. Wismueller, and Z. Balaton et al. Performance tools for the grid: State of the art and future. Technical report, APART WP3, 2004.
- [5] Network Working Group. eXternal Data Representation (XDR), August 1995. IETF RFC 1832.
- [6] A. Iosup and S. Vialle. Mobile robot navigation and self-localization system: Parallel and distributed experiments. In *The Dagstuhl Workshop on Plan-Based Control of Robotic Agents, Dagstuhl, Germany*, June 2003.
- [7] P. Kacsuk. Parallel program development and execution in the grid. In *IEEE International. PARELEC'02, Warsaw, Poland*, pages 131–141, September 2002.
- [8] K. Krauter, R. Buyya, and M. Maheswaran. A taxonomy and survey of Grid resource management systems for distributed computing. *Software Practice and Experience*, 32(2):135–164, February 2002.
- [9] M. Massie, B. Chun, and D. Culler. The Ganglia distributed monitoring system: Design, implementation, and experience. *Parallel Computing*, 30(7):817–840, July 2004.
- [10] Zs. Nemeth, G. Gombas, and Z. Balaton. Performance evaluation on Grids: Directions, issues, and open problems. In *Proceedings of the 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP'04), A Coruna, Spain*, pages 290–297. IEEE Computer Society Press, February 2004.
- [11] H.B. Newman, I.C. Legrand, P. Galvez, R. Voicu, and C. Cirstoiu. MonALISA: A distributed monitoring service architecture. In *CHEP 2003, La Jola, California*, March 2003.
- [12] F. Sabatier, A. De Vivo, and S. Vialle. Grid programming for distributed remote robot control. *International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE-2004)*, June 2004. (to appear).
- [13] W. Stallings. *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2 (3rd Ed.)*. Wiley Publishing House, 1998. ISBN: 0-201-48534-6.
- [14] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, and M.F. Kaashoek. CHORD: A scalable peer-to-peer lookup protocol for Internet applications. *IEEE/ACM Transactions on Networking*, 11:17–32, 2003.
- [15] N. Tapus, V. Cristea, M. Burcea, and V. Staicu. RoGrid towards a Romanian computational Grid. In *Proceedings of the 14th International Conference on Control Systems and Computer Science (CSCS14), Romania*, July 2004.
- [16] N. Tapus, E. Slusanschi, and T. Popescu. Distributed rendering engine. In *Proc. of the NATO Advanced Research Workshop on Adv. Environments, Tools, and Applications for Cluster Computing*, pages 207–215. Springer-Verlag, 2002.
- [17] B. Tierney, R. Aydt, D. Gunter, W. Smith, V. Taylor, R. Wolski, and M. Swany. White paper: A grid monitoring service architecture, February 2001. GGF Performance WG, Grid Working Document GWD-GP-6-1.
- [18] R. van Renesse, K. Birman, and W. Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Transactions on Computer Systems*, 21(2):164–206, May 2003.
- [19] J.S. Vetter and D.A. Reed. Real-time performance monitoring, adaptive control, and interactive steering of computational grids. *The International Journal of High-Performance Computing Applications*, 14:357–366, Winter 2000.
- [20] R. Wolski. Experiences with predicting resource performance on-line in computational grid settings. *ACM SIGMETRICS Performance Evaluation Review*, 30(4):41–49, 2003.
- [21] H. Yu and A. Vahdat. Design and evaluation of a conit-based continuous consistency model for replicated services. *ACM Transactions on Computer Systems*, 20(3):239–282, 2002.