

Streaming Grid: A Proposal for Grid Middleware Services Supporting Streaming Applications

Bikash Agarwalla
Ph.D. Student
College of Computing
Georgia Institute of Technology
801 Atlantic Drive NW, Atlanta, GA 30332-0280, USA
Email: bikash@cc.gatech.edu

1 Introduction

Advances in sensing capabilities, and computing and communication infrastructures are paving the way for new and demanding applications. Video-based surveillance, emergency response, disaster recovery, habitat monitoring, and telepresence are examples of such applications. These applications in their full form are capable of stressing the available computing and communication infrastructures to their limits. *Streaming applications*, as we refer to such applications, have the following characteristics: (1) they are continuous in nature, (2) they require efficient transport of data from/to distributed sources/sinks, and (3) they require the efficient use of high-performance computing resources to carry out compute-intensive tasks in a timely manner.

The focus of this challenge is in addressing the third component of the above characteristics, namely, the use of high-performance computing (HPC) resources to carry out compute-intensive tasks. Consider for example, a video-based surveillance application. The compute intensive part of such an application may consist of analyzing multiple camera feeds from a region to extract higher level information such as “motion”, “presence or absence of a human face”, or “presence or absence of any kind of suspicious activity”. Such an application can be represented as a coarse-grain dataflow graph, wherein the nodes represent increasing sophistication of computations that may need to be performed on the data stream to facilitate the extraction of high-level information.

We present a proposal for *Streaming Grid* consisting of middleware services supporting streaming applications. In particular, we present resource allocation system architecture for *Streaming Grid*. Condor [4] is a well-studied resource allocator for grid computing framework. Using Condor, we have built a scheduler called *E-Condor* for scheduling streaming applications in the current grid framework. *E-Condor* serves as a baseline for streaming application scheduler in grid environment. Before describing in detail our challenge and solution methodology, we present a brief overview of our background to put this proposal in perspective.

2 Background

We consider an ambient computing infrastructure consisting of heterogeneous resources such as sensor network, gateways, and grid as shown in Figure 1. To fulfill the quality of service requirements of streaming applications, on-demand access to this infrastructure is needed. We elaborate on our experiences and current work relating to different components of the infrastructure in order to set the stage for this proposal.

2.1 Application

In prior work [17], we have surveyed a variety of subsystems designed to be building blocks from which sophisticated infrastructure for ubiquitous applications can be assembled. Our work resulted in *UbiqStack*, a five-class taxonomy of system

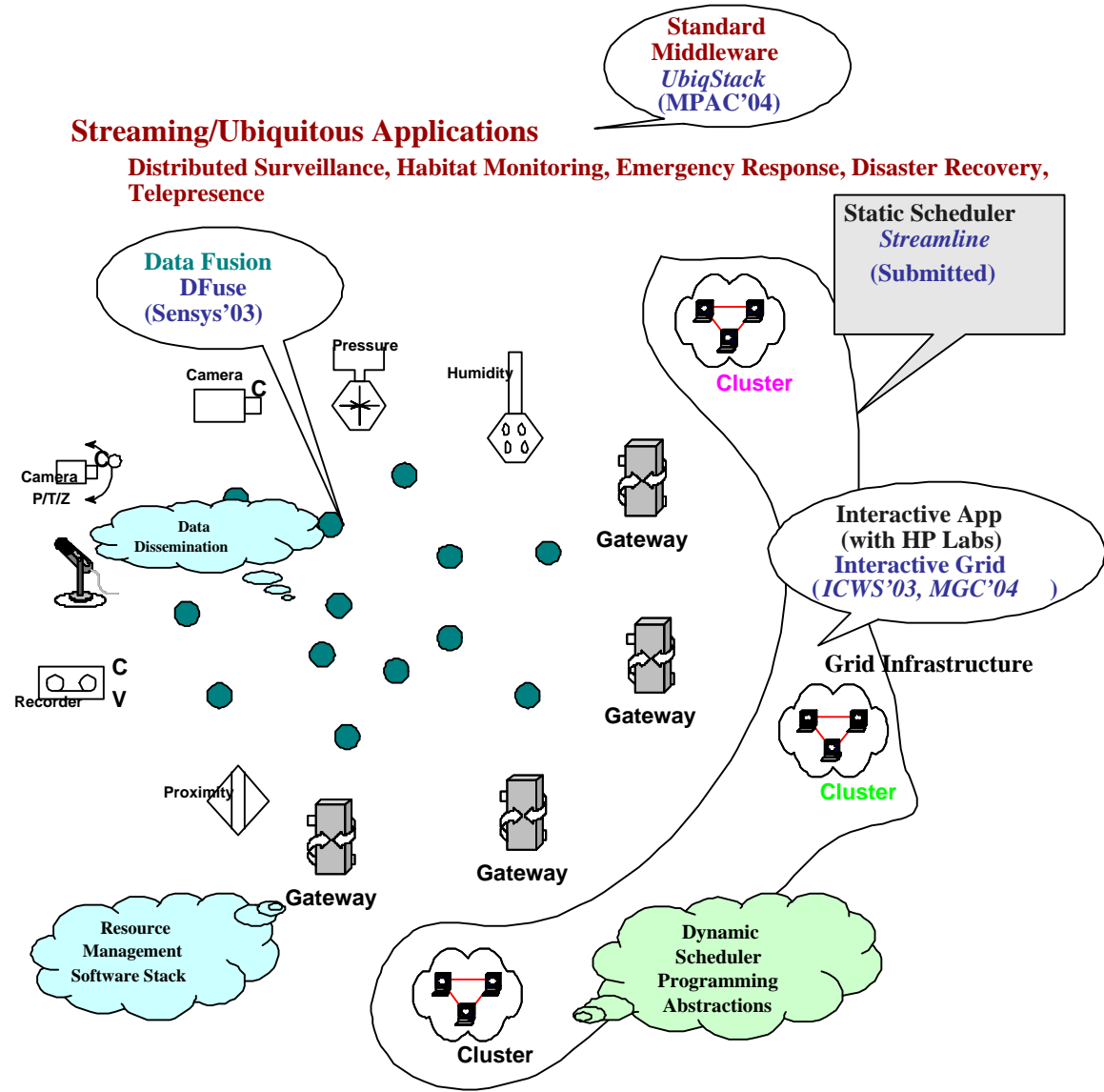


Figure 1. Ambient Computing Infrastructure

infrastructure components that can be used as “lego-pieces” for middleware services. The five-class taxonomy consists of *data storage and streaming*, *service and subscription*, *computation sharing*, *context management*, and *registration and discovery*. Out of these five classes, grid services enable *computation sharing* by providing access to remote computational resources on demand.

2.2 Sensor Network

In a streaming application, the data sources (such as cameras, microphones, temperature, pressure, and humidity sensors) are geographically distributed in different administrative boundaries. A sensor network provides the necessary infrastructure for transporting the streaming data to HPC resources. Traditional sensor network research has focused on issues such as data dissemination, routing, and network protocols taking into account the limited resource capabilities of sensor nodes. Our hypothesis is that by making use of ambient HPC resources, sophisticated time-critical streaming applications can be developed. Thus, our sensor network infrastructure consists of sensors, actuators, resource constrained *relay nodes* with data transport and limited computational capabilities, and more capable *gateways* for providing access to HPC resources. In DFuse [14], we have presented data fusion abstractions whereby sophisticated processing can happen within the sensor network. We are currently exploring data dissemination challenges in the presence of gateways and HPC resources.

2.3 Gateways

Gateways fall between sensor nodes and the HPC resources in their computation, communication, and power capabilities. For instance, in a campus surveillance application, IPAs carried by students throughout the campus can act as gateways providing access to grid resources across different departments. Gateways function as proxies for grid resources performing data transport to/from the sensor network. Because of their unique role, gateways require a software stack capable of supporting multiple protocols. In this context, we are looking at providing lightweight grid services for gateway nodes. We are also exploring resource management issues related to gateways.

2.4 Grid Infrastructure

Grid computing has traditionally focused on scientific and engineering applications supporting batch jobs. In prior work [3, 24], we have extended grid architecture to support *interactive* applications such as CAD/CAM and office applications (MS Outlook, MS Access). Through Interactive Grid, users of these applications gain access to remote desktop sessions in a grid environment.

We are currently exploring issues related to programming abstractions for Streaming Grid. Programming abstractions for streaming applications in a dedicated cluster environment have been developed in the past [18, 2, 20, 16, 22]. Due to the dynamism and heterogeneity in resources, grid poses new challenges for these programming abstractions, possibly requiring new abstractions. We are also designing scheduling services in Streaming Grid capable of static placement of streaming applications and dynamic adaptation depending on the resource availability and application requirements. In the rest of this writeup, we focus on our scheduling service for streaming applications using an existing grid scheduler.

3 Challenge: Scheduling Problem

The streaming application in our scheduling problem is represented by a coarse-grain directed acyclic dataflow graph. Each node of the dataflow graph represents a continuously running application stage with the direction of dataflow denoted by the edges. The scheduling system allocates resources to the stages of the application dataflow graph so as to meet the quality of service requirements such as latency and throughput.

At some level such coarse-grain dataflow graphs resemble task-graphs that have been the focus of multiprocessor scheduling work from the 70's [1, 7, 21, 13, 23, 12, 15, 25]. However, in multiprocessor scheduling, a task-graph (a directed acyclic graph) is used as an ordering mechanism to show the dependencies among the individual tasks of a parallel computation. These dependencies are respected and exploited in arriving at a mapping heuristic (since the scheduling problem is known to be NP-Complete) that maximizes the utilization of the computational resources and reduces the completion time of the application. The coarse-grain dataflow graph of a streaming application, on the other hand, is a representation of the processing that is carried out on the data during its passage through the pipeline of stages. In the steady state, all the stages of the pipeline are working on different snapshots of the stream data. For example in a video-based surveillance application, when the n-th

stage is working on the (hitherto transformed results of the) i^{th} frame of video, the first stage of the pipeline is working on the $(n + i)^{th}$ frame. So the scheduling of such streaming applications is not an ordering issue; rather, it is a matter of mapping the different stages of the pipeline to the available resources respecting the computation and communication requirements of each stage with a view to optimizing the latency and throughput metrics of the entire pipeline.

Scheduling in grid has primarily focused on providing support for batch-oriented jobs (See Nabrzyski, et al. [19] for a survey of current grid schedulers.) A wide variety of meta-schedulers and resource brokers using Globus Toolkit [10] have been developed by other research projects such as Condor[4], Legion[6], and Nimrod-G [5]. Through our E-Condor architecture presented in the next section, we show how these batch schedulers can be adapted for streaming applications.

4 Solution: E-Condor Architecture

We have selected Condor [4] as an example batch scheduler, due to its maturity and flexibility, as a vehicle for adapting an existing grid scheduler to support streaming applications.

Condor uses DAGMan[11] to launch applications that are specified by a task-graph, DAGMan is designed for task-graph based batch jobs with data dependencies and hence launches a task only after all its predecessors have *finished* execution. However, as we have observed before, in a streaming application, each stage of the dataflow graph is concurrently working on a snapshot of the continuous stream data. Therefore, we have developed a simple stream scheduler on top of Condor called *E-Condor*. E-Condor uses Condor to obtain the resources necessary to launch the individual stages of a streaming application. But prior to launching, E-Condor takes care of setting up all the necessary coupling between the stages commensurate with the dataflow graph of the application.

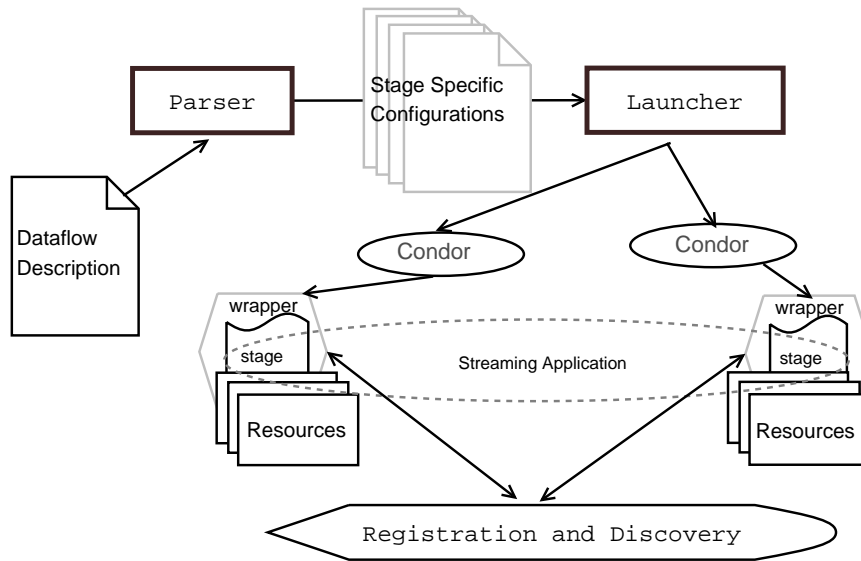


Figure 2. E-Condor Architecture

Figure 2 shows the architecture of E-Condor, which has the following components:

- A *parser* that automatically generates the entire dataflow graph and the per-stage configuration files given a high-level description of the streaming application.
- A *launcher* that uses Condor to map the stages of the dataflow graph to different computational nodes provided by Condor.
- A *registration and discovery* service for establishing the predecessor/successor relationships among the stages of the dataflow graph. The architecture automatically generates wrapper code for each stage to register itself with this service,

determine its predecessors and successors using the per-stage configuration file, and establish the necessary connections to them.

- A *synchronization protocol* that ensures that all the stages have established the necessary connections to one another before actually starting the application-supplied code for that stage.

E-Condor serves as a baseline scheduler for streaming applications on the grid using an existing grid scheduler. In the next section, we present our system architecture that integrates E-Condor with existing grid middleware.

5 System Architecture

Our system architecture is guided by following design goals: (i) For quicker deployment, the system should make use of existing grid functionalities as much as possible (ii) The resource allocation system should function in a dynamic environment where resource availability and node connectivity change frequently.

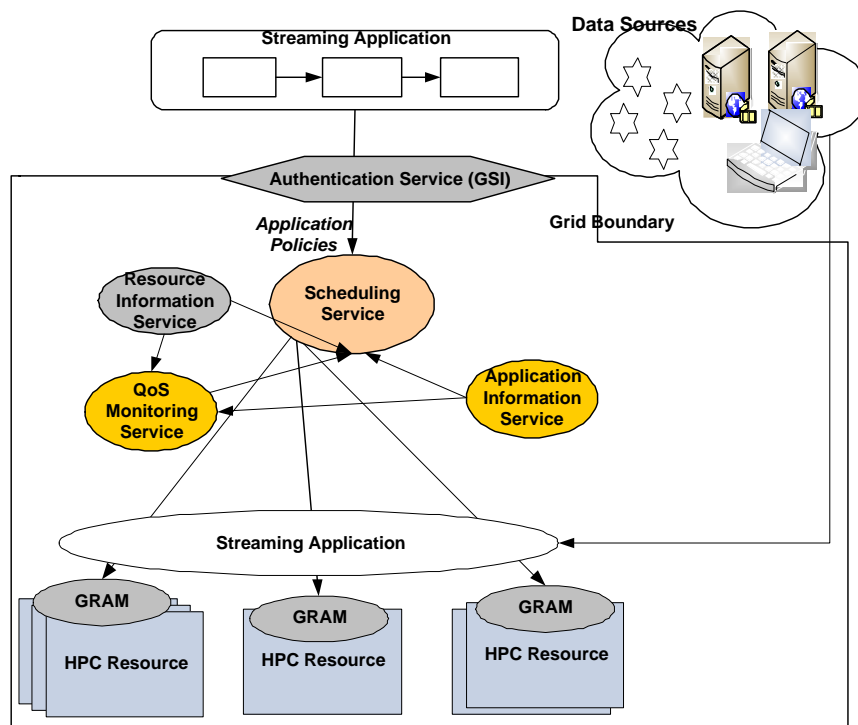


Figure 3. Resource Allocation System Architecture

Figure 3 shows our resource allocation system architecture. The user first authenticates to the grid using Grid Security Infrastructure(GSI). Once authenticated, the user submits the streaming application coarse-grain dataflow graph to the scheduling service. The scheduling service takes into account the current resource availability (through resource information service [8, 26]) and existing applications running in the grid (through application information service) in deciding whether to launch a new instance of the application. Once resources have been allocated, the application dataflow graph is instantiated on individual resources using Grid Resource Allocation and Management(GRAM) [9] system. The QoS monitoring service is responsible for monitoring the quality of service requirements of the application and dynamically adapting resource assignment by contacting the scheduler. The QoS monitoring service infers the computational requirement of the running application by periodically contacting the application information service and the resource information service.

The authentication service(GSI), resource information service [8] and GRAM [9] are part of the present Globus Toolkit [10] infrastructure. Scheduling, application information, and QoS monitoring are additional grid services that are introduced by our middleware system architecture. In this challenge, we have focused on the scheduling service, and have implemented E-Condor.

6 Conclusion

We have proposed *Streaming Grid*, a set of middleware services supporting streaming applications in a grid environment. Our *E-Condor* architecture enables resource allocation for streaming applications using an existing grid scheduler Condor [4]. We have implemented *E-Condor* in the Globus Toolkit [10]. Our scheduling service provides a rich set of facilities that include: (i) APIs for users to specify the resource requirements of each stage and the dependencies among the stages of the streaming application; (ii) APIs for job submission that allows multiple applications to be submitted to the scheduler at the same time; each application receives a unique name; and (iii) APIs for querying job status using the unique name. We have learned that while *E-Condor* suffices as a baseline scheduler, new scheduling algorithms may need to be developed specifically for streaming application to improve performance.

Our *Streaming Grid* proposal is a step towards middleware services that integrate ambient computing infrastructure spanning sensor network, gateways, and grid resources. We plan to explore the appropriate programming abstractions and dynamic scheduling services in *Streaming Grid* as part of future work.

References

- [1] T.L. Adam, K. Chandy, and J. Dickson. A comparison of list schedules for parallel processing systems. *Communications of the ACM*, 17(12):685–690, Dec 1974.
- [2] Sameer Adhikari, Arnab Paul, and Umakishore Ramachandran. D-stampede: Distributed programming system for ubiquitous computing. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS)*, Vienna, July 2002.
- [3] Sujoy Basu, Vanish Talwar, Bikash Agarwalla, and Raj Kumar. I-GASP: Interactive grid environment provided by application service providers. In *1st International Conference on Web Services (ICWS'03)*, Las Vegas, USA, 2003.
- [4] R. Boer. Resource management in the Condor system. Master's thesis, Delft University of Technology, 1996.
- [5] Rajkumar Buyya, David Abramson, and Jonathan Giddy. Nimrod/g: An architecture for a resource management and scheduling system in a global computational grid. *High Performance Computing (HPC) ASIA*, 2000.
- [6] Steve J. Chapin, Dimitrios Katramatos, John Karpovich, and Andrew S. Grimshaw. The Legion resource management system. In *Job Scheduling Strategies for Parallel Processing*, pages 162–178. Springer Verlag, 1999.
- [7] E.G. Coffman. *Computer and Job-Shop Scheduling Theory*. Wiley, New York, 1976.
- [8] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing. In *Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10)*, August 2001.
- [9] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke. A resource management architecture for metacomputing systems. In *IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing*, pages 62–82, 1998.
- [10] Ian Foster and Carl Kesselman. Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, Summer 1997.
- [11] James Frey, Todd Tannenbaum, Ian Foster, Miron Livny, and Steven Tuecke. Condor-G: A computation management agent for multi-institutional grids. In *Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC10)*, pages 55–63, San Francisco, CA, August 2001.
- [12] A. Gerasoulis and T. Yang. A comparison of clustering heuristics for scheduling dags on multiprocessors. *Journal of Parallel and Distributed Computing*, 16(4):276–291, Dec 1992.
- [13] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnoy Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

- [14] Rajnish Kumar, Matthew Wolenetz, Bikash Agarwalla, JunSuk Shin, Phillip Hutto, Arnab Paul, and Umakishore Ramachandran. DFuse: A framework for distributed data fusion. In *Proceedings of ACM SenSys 2003.*, Los Angeles, CA, USA, November 2003.
- [15] Yu-Kwong Kwok and Ishfaq Ahmad. Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors. *IEEE Transactions on Parallel and Distributed Systems*, 7(5):506–521, 1996.
- [16] Tobin J. Lehman, Stephen W. McLaughry, and Peter Wycko. T spaces: The next wave. In *Hawaii International Conference on System Sciences (HICSS'99)*, 1999.
- [17] Martin Modahl, Bikash Agarwalla, T. Scott Saponas, Gregory Abowd, and Umakishore Ramachandran. Towards a standard ubiquitous computing framework. In *2nd International Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC'04)*, Toronto, Canada, Oct 2004.
- [18] Martin Modahl, Ilya Bagrak, Matthew Wolenetz, Phillip Hutto, and Umakishore Ramachandran. Mediabroker: An architecture for pervasive computing. In *Proceedings of the 2nd IEEE International Conference on Pervasive Computing and Communications*, March 2004.
- [19] Jarek Nabrzyski, Jennifer M. Schopf, and Jan Weglarz. *Grid Resource Management: State of the Art and Future Trends*. Kluwer Academic Publishers, Sep 2003.
- [20] Umakishore Ramachandran, Rishiyur S. Nikhil, Nissim Harel, James M. Rehg, and Kathleen Knobe. Space-time memory: A parallel programming abstraction for interactive multimedia applications. In *Principles Practice of Parallel Programming*, pages 183–192, 1999.
- [21] C.V. Ramamoorthy, K.M. Chandy, and M.J. Gonzalez. Optimal scheduling strategies in a multiprocessor system. *IEEE Trans. on Computers*, C-21(2):137–146, Feb 1972.
- [22] S. Microsystems. Javaspaces specification, March 1998.
- [23] G.C. Sih and E.A. Lee. A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures. *IEEE Trans. on Parallel and Distributed Systems*, 4(2):75–187, Feb 1993.
- [24] Vanish Talwar, Bikash Agarwalla, Sujoy Basu, and Raj Kumar. Architecture for resource allocation services supporting interactive remote desktop sessions in utility grids. In *2nd International Workshop on Middleware for Grid Computing (MGC 2004)*, Toronto, Canada, Oct 2004.
- [25] Haluk Topcuoglu, Salim Hariri, and Min-YouWu. Task scheduling algorithms for heterogeneous processors. In *Proceedings of the 8th Heterogeneous Computing Workshop*, pages 3–14, 1999.
- [26] Rich Wolski. Dynamically forecasting network performance using the network weather service. *Journal of Cluster Computing*, 1:119–132, January 1998.