

# Resource Aggregation and Workflow with Webcom

Oisín Curran<sup>1</sup>, Paddy Downes<sup>1</sup>, John Cunniffe<sup>1</sup>, Andy Shearer<sup>2</sup>,  
and John P. Morrison<sup>3</sup>

<sup>1</sup> Dept of Information Technology, National University of Ireland, Galway,  
Galway, Ireland

oisin.curran@geminga.it.nuigalway.ie, {paddy,jc}@it.nuigalway.ie

<sup>2</sup> Dept of Physics, National University of Ireland, Galway,  
Galway, Ireland

andy.shearer@nuigalway.ie

<sup>3</sup> Centre for Unified Computing, University College Cork,  
Cork, Ireland

j.morrison@cs.ucc.ie

**Abstract.** Efficient exploitation of the aggregate resources available to a researcher is a challenging and real problem. The challenge becomes all the greater when researchers who collaborate across functional or administrative domains need to pool their disjoint heterogeneous resources to achieve their objectives. Support tools are becoming available for these ad hoc resource integration and sharing scenarios. The focus of this paper is the identification of suitable deployment and usage strategies when using the workflow approach with these tools. In particular, we present a novel two-level peer-to-peer model for dynamic resource aggregation that operates entirely at the user level. This sidesteps the need for system level middleware and administrative support. This paper presents our strategy, the underlying framework and the workflow expression and evaluation semantics.

## 1 Introduction

Much research is conducted in environments where there exist multiple independent resources and multiple dynamic groups of occasionally collaborating users with varying levels of access to subsets of those resources. Resources may be widely distributed and have very different architectures and administrative policies. Such environments evolve naturally within and among academic and industrial research facilities. These sites are typically funded departmentally yet collaborations may span departmental or other administrative or functional borders.

Without a large-scale, ongoing and predictable pattern of collaboration there is little motivation to implement and maintain a computational grid [1] solution to the problem of resource aggregation. Even if such a solution were provided it may not be optimal for many classes of problem; the grid model of distributed computing is largely focused on high throughput batch processing and not easily tuned to support other priorities such as deadline or interactive (steered) processing.

In such scenarios there is a need for lightweight user-level tools that facilitate dynamic resource aggregation while ensuring the efficient use of resources. Such tools are emerging, yet the availability of tools that facilitate resource aggregation is not in itself enough to guarantee efficient resource usage and sharing.

This report presents our use of the Webcom system [2], [3], [4] to aggregate and abstract distributed heterogeneous resource sets. These resources are typically under the control of different administrators and in active use by different research groups for a diverse range of application types. We demonstrate our use of Webcom as a tool for resource aggregation using a novel two-level peer-to-peer *Overlay Metacomputer* model.

This paper makes the following contributions,

1. We propose a methodology for the dynamic aggregation of disjoint independent resources. The aggregate resource set behaves as a distributed virtual machine that interprets a common workflow expression format on all platforms, thereby abstracting site differences. Our model provides a locality-aware peer based system for load balanced workflow execution. This approach relies solely on user-level access to resources and does not depend on any particular middleware or uncommon site policies.
2. We present our workflow expression and execution tools built on top of the Webcom graph evaluation engine. With these tools we can make use of Webcom's unique semantics and task distribution capabilities to achieve efficient workflow evaluation.
3. We present results of tests of our model, using both synthetic and real application examples in a real and very heterogeneous environment consisting of distinct and independently administered resources of different architectures with only partially overlapping user communities.

The rest of this paper is organised as follows. In section 2 some background details are presented to put our current report in context. The approach taken is detailed in section 3. A motivating example is presented and discussed in section 4. Initial results of experimental evaluations are presented in section 5. Related work is discussed in 6. We close this report by presenting conclusions and outlining future work targets in section 7.

## 2 Background

Webcom is based on the Condensed Graph (CG) model of computing [5]. A CG is a directed acyclic graph (DAG) with enhanced semantics. Each node in a CG is an executable entity. A node may represent a single instruction or an entire graph that has been *condensed* (abstracted) to node form. Certain nodes affect the flow of control (e.g., the conditional node) and, used in conjunction with CG edge semantics, direct the order of graph evaluation. The CG model is intended to give the programmer the benefits of both the control flow and data flow models of computing in a single system. Webcom is implemented as a peer-based CG evaluation engine; a distributed virtual machine that interprets

the CG language. A detailed description of Webcom is presented in [4]. Webcom is implemented entirely in Java to increase its portability. Our work focuses on implementing workflow via Webcom, and analysing suitable deployment strategies for different application and environment types, to support the development of real applications with Webcom.

### 3 Approach

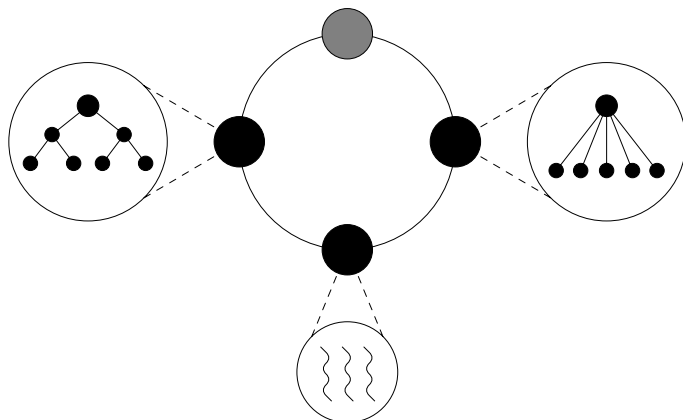
This section presents the runtime organisation and workflow expression strategies of our model. These have been designed to leverage the capabilities of the Webcom framework and the features of the CG model to support application deployment in heterogeneous computing environments.

#### 3.1 Runtime Topologies

Our approach to resource aggregation consists of constructing a two level overlay network across the available independent resources. When combining mixed resources including stand-alone machines, Condor pools [6], [7], and clusters controlled in space sharing mode, a top level ring links the sites. This is in keeping with the work presented in [8] where it was shown that a peer-to-peer ring of Condor pools proved an efficient, scalable and fault tolerant method to connect sites. This top level ring is set up by starting the Webcom service at each site using the site's task submission interface. These Webcom instances connect to each other forming the top level *Webcom world*.

Systems have been reported in the literature that use Ring (e.g., Chord [9] and Tapestry [10]), and Balanced Tree [11] based topologies. The Webcom system is in this sense 'model free'; machines can be connected in any of these structures or combinations thereof. We exploit this fact to group peers in the topology best suited to their characteristics. Overlay strategies for connection management are well established, with many efficient and robust solutions reported in the literature [12], [11], [13], so new topologies are not a strong focus of our current research. Hence, our illustrative example uses a ring at the top level, rather than any more sophisticated connection management structure.

Similarly, a site-internal Webcom world is set up at each location using the processors allocated by the local resource manager as appropriate. The topology chosen at each site depends on the nature of the site and the number of processors to be used. In a mixed and widely disbursed or very heterogeneous Condor pool, a tree topology is suitable as this means less traffic over the network and less impact to the Webcom world when an individual Webcom is preempted by Condor [14], [11]. At a dedicated space sharing site where preemption is not a factor and all processors are connected to the same switch, a star topology is more efficient. There remains the risk of losing connection with the entire cluster, but the loss of an individual node is far less likely than at a cycle scavenging site. Webcom can also be used to group any number of independent workstations as it does not rely on the existence of a batch job manager for resource access.



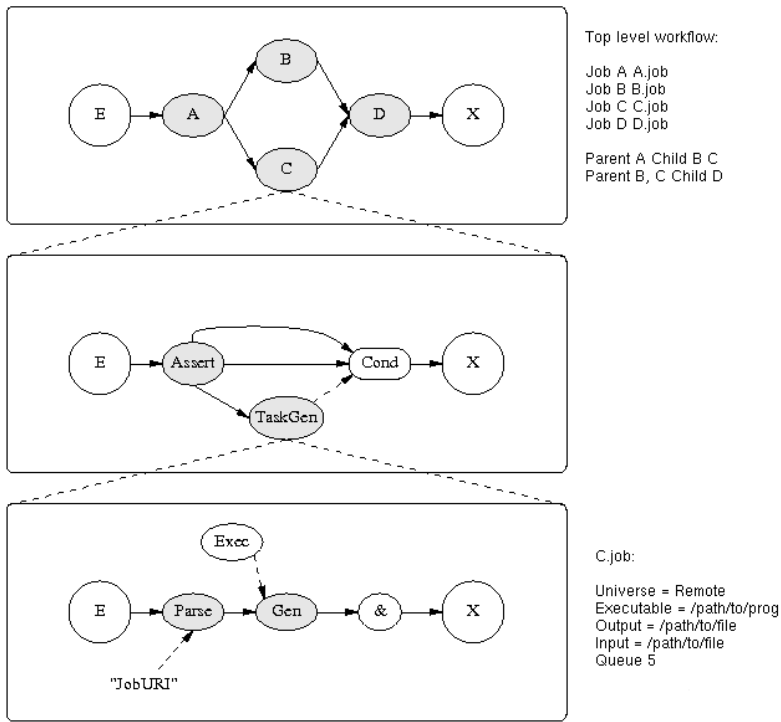
**Fig. 1.** An example of a Webcom overlay metacomputer. This example scenario shows the aggregation of resource from three sites: a Condor pool shown on the left, a PBS queue on the right, and an SMP machine at the bottom. A top level ring is formed with these and a stand-alone workstation (at the top of the circle). The latter may act as the point of submission for the user’s workflow and also contribute to task execution. With this overlay in place, workflow graphs can be evaluated across the aggregate resource set, with Webcom providing load balancing across the individual sites and abstracting underlying differences in platform and topology.

A schematic of the approach is presented in figure 1. This shows a scenario wherein three sites form a top level Webcom world. Each site hosts an independent Webcom world.

### 3.2 Workflow and Job Description Languages

Expression languages for scientific workflow have been presented in a range of projects for various systems [15], [16], [17]. Our workflow and job description languages make use of the Condor DAGMan syntax and offer the potential to go beyond its semantics. DAGMan workflows and CGs are both DAG based, therefore every DAGMan workflow has an equivalent CG representation. The workflow structure, expressed in the DAGMan language as a collection of jobs in a parent/child hierarchy maps to an equivalent shape CG. As Webcom facilitates flow control with conditional execution, iteration, and explicit recursion, it can be seen as a superset of the DAGMan language. Mapping the semantic concepts of Condor/DAGMan to the Webcom equivalent, as the two systems have radically different deployment architectures, requires the use of specific Webcom *nodes* designed to handle particular Condor job types. Figure 2 shows a synthetic example application.

Our Webcom based workflow strategy uses three layers of condensation in each workflow. The top layer is the workflow shape; the middle layer provides exception handling. This is built into the graph expression making exception handling a fully distributed part of workflow execution. The lowest layer shows



**Fig. 2.** Workflows are implemented as a set of graphs with this pattern of condensation. The top level graph expresses the logical sequence of tasks in the workflow. The middle layer provides exception handling using Webcom’s flow control operators and lazy evaluation semantics. The lowest level graph uses Webcom’s ability to add nodes to a graph at runtime, using an enumeration node, dynamically creating a cluster of one or more jobs.

the mapping from workflow task expression (attribute/value pairs) to graph. When one of these graphs is executed on a Webcom worker, the “Parse” node downloads the job definition from the submission site and generates a set of parametrised job entities. These are passed to the “Gen” node who generates a set of work execution nodes (instances of “Exec”), feeding each its own version of the job description. This allows for site specific and task specific information to be calculated when and where appropriate. This deferred approach to task interpretation means that task definitions can change after submission of the initial workflow. In the event of a task’s failure, its children must not attempt to execute and the workflow should be aborted. In the absence of a central point of control (fully decentralised workflow execution), where the tasks have been submitted to the workflow system and are pooled for execution, conditional execution of child nodes is used. The condition for execution of a child node is the successful completion of all parents. Failure of a parent results in the propagation of trace data through the graph that facilitates fault detection and

re-running of the outstanding tasks. This behaviour is guaranteed by the lazy evaluation semantics of the CG model. The nodes containing the task execution logic are only evaluated when needed. If all parents are not fully successful, the child task is not needed and so needless resource consumption is minimised.

## 4 Example Application

As part of the Webcom project we are collaborating with the Medical Physics department of the University College Hospital Galway (UCHG) in an ongoing effort to develop a distributed Radiation Treatment Planning (RTP) workflow solution based around the use of a Monte Carlo (MC) radiation transport simulation package.

The use of the MC technique in medical physics has become increasingly common. In radiotherapy, it is widely accepted that MC is the most accurate and general calculation method available [18]. A number of review articles exist on this topic [19], [20], [21]. The main drawback of this method is the time involved. The more traditional, but less accurate [22], solutions such as the pencil beam [23] or collapsed cone [24] calculation will take minutes to perform on a standard desktop. A MC calculation can take days to complete. Our objective is to provide MC based results in a time-frame that makes MC a viable option for clinical use.

The full workflow for this application involves data transport, the acquisition and generation of simulation model input data, preparation of executables for distribution based on currently available resources, the simulation itself, result accumulation and verification, format translation, and visualisation. This application and our approach to it are described fully in a forthcoming paper. The purpose of this overview is to demonstrate the applicability of our resource aggregation approach and workflow model to very real and computationally intensive problems.

### 4.1 Application Requirements

The main computational step in the RTP workflow is the MC phase. However, there are significant data input and output issues. An accurate run of the simulation results in the transport, analysis and visualisation of approximately 20 GB of data. Runtimes are dependent on a number of user specified parameters; a usable result can be generated in 70 to 100 CPU hours. A clinical time-frame is in the order of one to three hours. Providing medical rather than computational science experts with the necessary computational power in a flexible, reliable and accessible manner is the central challenge of this application. Our approach answers that challenge by facilitating the harnessing of disjoint resources *on-demand*, and ensuring the efficient use of those resources.

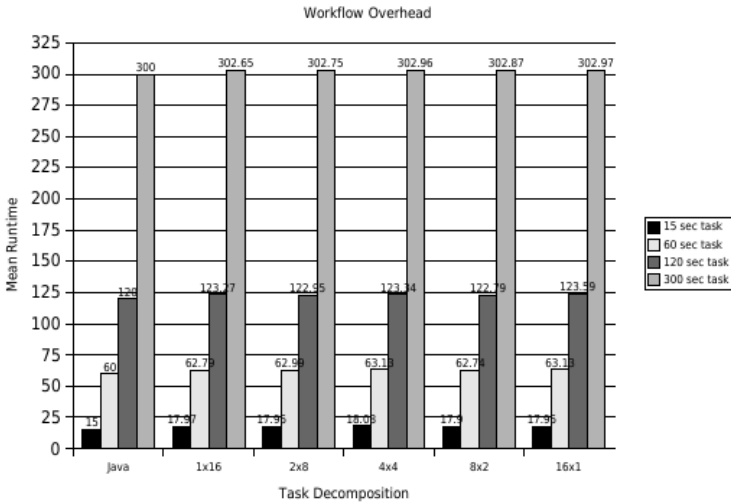
### 4.2 Application Environment

Due to the interdepartmental nature of our research group, a mixed collection of resources are accessible for use. These include a 64 processor cluster (32 dual Intel Xeon 2.4 GHz processor machines with two GB of RAM per machine)

managed by Torque [25] [26] and Maui [27] in space sharing mode, a very heterogeneous Condor pool featuring high end graphics workstations (two dual-core AMD Opteron processors with 16GB of RAM), more modest Linux workstations, and general purpose laboratory machines (Intel Pentium III and Pentium IV based machines with between 128MB and 384MB of RAM running Windows 2000 or Windows XP). Added to this there is a 16 CPU AMD Opteron SMP machine with 32GB of RAM, partially managed by Torque/Maui and donating spare cycles to the Condor pool. Other independent (typically midrange) workstations are managed directly as Webcom clients.

### 4.3 Discussion

None of these resources alone can provide the computational needs of the application due to queue lengths, policy restrictions, or a combination of these and other factors at individual sites. The benefit of our approach is the ease and consistency with which the available resource subsets at each site can be harnessed into an efficient and powerful metacomputer suitable for the processing of the application workflow. Users can drive the application from a portal that supports parametrisation of the workflow and the upload of patient- and treatment plan-specific inputs. Progress can then be steered based on the expert user's interpretation of intermediate results.

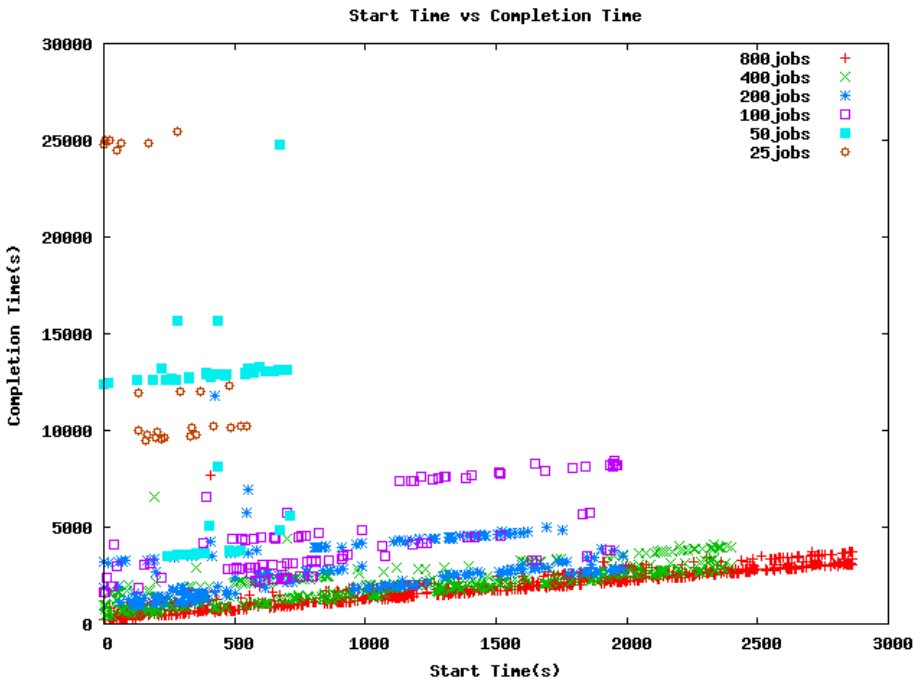


**Fig. 3.** A sequence of decompositions for a parallel application. The ‘Java’ timings show the basic performance of a 16 way parallel application. The ‘1x16’ arrangement refers to a one node workflow version of the application. The ‘2x8’ arrangement uses two nodes, each of which executes eight threads at the application level. The ‘16x1’ set of timings show the performance when the parallelism of the application is entirely exposed at the workflow level. In all cases, for all task durations, the cost of workflow expression is approximately two or three seconds.

## 5 Evaluation

The basic performance of Webcom as an efficient resource manager and graph execution engine is shown in figure 3. This experiment shows Webcom's performance when executing different workflow expressions of a parallel application. The test application simulates a scalable job capable of being decomposed into different arrangements. As the decomposition becomes increasingly fine grained with more of the parallelism exposed at the workflow level (rather than the application logic level), the overhead of graph management remains reasonably constant, with execution times as short as 15 seconds.

The RTP application is scalable and can be tuned dynamically to the available resources such that each node in the MC phase has approximately the same runtime. Figure 4 shows the results of executing the MC phase of the RTP workflow over a dynamically aggregated collection of 165 processors from different sites.



**Fig. 4.** A plot of start times to finish times for different decompositions of the radiotherapy workflow executed over a collection of 165 machines using our resource aggregation strategy to combine resources controlled by different managers including Condor and PBS, in conjunction with stand-alone machines running the Webcom service. The arrangement of 800 jobs shows the best performance as it is capable of making use of all available resources. The smaller decompositions expose less parallelism at the workflow level and consequently fewer machines are used, leading to longer execution times for each job and consequently longer workflow execution times.

This shows that the more fine grained decompositions of the task, which equate to a larger number of shorter jobs, results in better completion times. One reason for this is that the overall impact of jobs that are delayed on non-dedicated machines is greatly reduced as each individual job represents a smaller fraction of the overall workflow. The efficient handling of large parallel workflows over mixed machines, demonstrates the utility of our approach.

## 6 Related Work

This work builds on investigations conducted in many areas related to task and resource management. Pinchak [28] describes an approach to building a load balancing meta-queue over a set of independent batch processing sites using the concept of a *placeholder* task, inserted into each site's queue, whose purpose is to pull down instructions from a work server and execute tasks using the queue's resources. Our work is conceptually similar in that we use multiple independent sites to form a metacomputing platform by overlay. However, the work in [28] and [29] (the latter presents a DAG oriented extension to server) achieves load balancing via self-scheduling workers that pull tasks from a single master site following a fixed eager strategy. The Master/Worker (MS) model of work distribution, as used in [28], is popular and well suited to certain classes of problems. However it is not necessarily the most scalable model as the master's ability to dispatch work and collect results can prove a bottleneck [30]. The use of hierarchical MS topologies has been studied as a step towards dealing with the scalability issue [31]. A fundamental difference between [28] and this project is our use of the peer-to-peer model [32] to achieve greater scalability.

Other systems offer a single package solution for resource aggregation and workflow execution, e.g., the Java CoG kit [33] offers the ability to use a suite of resource access protocols including Globus [34], SSH [35], and web services, for workflow execution. Java CoG kit is built on the client/server model with pre-packaged components for interacting with various resource managers. Workflow is defined and managed using graphical or script languages that are translated to an XML-based intermediate representation. Our work differs as we use a multi-level P2P overlay abstraction for resource aggregation to increase scalability and robustness. In addition, as our approach is based on a graph-oriented model of computing that provides a strong theoretical basis from which to reason about the execution of workflow graphs. This model also provides conditional and iterative execution, recursive graph definitions, and combinations of lazy and eager evaluation strategies, which we leveraged to achieve decentralised exception handling.

The use of heterogeneous resources introduces the problem of platform dependence; we have tackled this problem using concepts developed within the Condor project [6], [7]. Condor supports cycle harvesting across heterogeneous non-dedicated resources to implement a distributed batch system focused on high throughput computing [36], [37]. Our technique uses the job and workflow description languages developed in Condor and incorporates ClassAd [38] matching with novel worker filtering techniques to achieve task to resource matching.

Support for the expression and correctly ordered submission of dependency constrained job collections, or workflows, is implemented in Condor via the DAGMan tool [39]. DAGMan is a user level tool whose purpose is to ensure that jobs are submitted in dependency-preserving order; it exerts no influence on Condor's placement and scheduling behaviour. Condor is batch-oriented and assumes that jobs are independent of each other; DAGMan proceeds as a Condor client, working on behalf of a user, submitting jobs when all their dependencies have been satisfied. Blythe et al. [40] discuss the impact of this lack of workflow awareness at the scheduler or task allocation level; excessive or redundant input and output transfer is identified as an issue with the Condor/DAGMan approach. Our approach is built on the Webcom graph evaluation engine and so the scheduler has access to all the inter-task dependency information in the workflow expression.

## 7 Conclusions and Future Work

In this paper we have described our approach and methodology for resource aggregation and collaboration support. The problem of integrating the diverse and distributed heterogeneous resources available to researchers has been tackled with a user-level overlay metacomputer. In addition, we have described the usage of our tools designed to support workflow execution in a dynamic peer-to-peer context. Initial results from both simulation and real applications have been presented to illustrate the efficiency of our model.

The purpose of this work is to support the development of solutions to real problems; these problems exhibit significant data and computational requirements. The solutions proposed herein are capable of exploiting the aggregate computing power of disjoint resource sets. Our future work will involve further optimisation of the tools discussed and support for dynamic task priority management, as well as enhancements to the automation of topological structuring and restructuring of Webcom overlays.

## References

1. Foster, I., Kesselman, C. (eds.): *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, Los Altos, CA (1999)
2. Morrison, J.P., Power, D.A., Kennedy, J.J.: Webcom: A volunteer-based metacomputer. *The Journal of Supercomputing* 18(1), 47–61 (2001)
3. Morrison, J.P., Power, D.A., Kennedy, J.J.: An evolution of the Webcom metacomputer. *The Journal of Mathematical Modelling and Algorithms* 23(2), 263–276 (2003) (Special Issue on Computational Science and Applications)
4. Morrison, J., Coghlan, B., Shearer, A., Foley, S., Power, D., Perrot, R.: WebCom-G: A candidate middleware for Grid-Ireland. *International Journal of High Performance Computing Applications* 20(3), 409–422 (2006)
5. Morrison, J.P.: *Condensed Graphs: Unifying Availability-Driven, Coercion-Driven and Control-Driven Computing*. PhD thesis, Technische Universiteit Eindhoven (1996)

6. Litzkow, M., Livny, M., Mutka, M.: Condor - a hunter of idle workstations. In: Proceedings of the 8th International Conference of Distributed Computing Systems (June 1988)
7. Thain, D., Tannenbaum, T., Livny, M.: Distributed computing in practice: the Condor experience. *Concurrency - Practice and Experience* 17(2-4), 323–356 (2005)
8. Butt, A.R., Zhang, R., Hu, C.Y.: A self-organising flock of Condors. *Journal of Parallel and Distributed Computing* 66, 145–161 (2006)
9. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: Proceedings of the ACM SIGCOMM 2001 Conference, San Diego, California (August 2001)
10. Zhao, B.Y., Stribling, L.H., Rhea, J., Joesph, S.C., Kubiatowicz, A.D.J.D.: Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications* 22(1) (2004)
11. Celaya, J., Arronategui, U.: Scalable architecture for allocation of idle CPUs in a P2P network. In: Gerdnt, M., Kranzlmüller, D. (eds.) *HPCC 2006*. LNCS, vol. 4208, pp. 240–249. Springer, Heidelberg (2006)
12. Stioica, I., et al.: Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw.* 11(1), 17–32 (2003)
13. Rowstron, A., Druschel, P.: Pastry: scalable, decenteralized object location and routing for large-scale peer-to-peer systems. In: Proceedings of the Middleware 2001, IFIP/ACM International Conference on Distributed Systems Platforms, pp. 329–350 (2001)
14. Jagadish, H., Ooi, B., Vu, Q.: Baton: A balanced tree structure for peer-to-peer networks. In: Proceedings of the 31st VLDB Conference, pp. 661–672 (2005)
15. Fahringer, T., Qin, J., Hainzer, S.: Specification of grid workflow applications with AGWL: An abstract grid workflow language. In: *EEE International Symposium on Cluster Computing and the Grid 2005 (CCGrid 2005)* (May 2005)
16. Alt, M., Hoheisel, A., Pohl, H.W., Gorlatch, S.: Using high level petri-nets for describing and analysing hierarchical grid workflows. In: Proceedings of the CoreGRID Integration Workshop 2005 (2005)
17. Hoheisel, A., Der, U.: An XML-based framework for loosely coupled applications on grid environments. In: Sloot, P., et al. (eds.) *International Conference on Computational Science*, pp. 245–254. Springer, Heidelberg (2003)
18. Mohan, R.: Why Monte Carlo? In: Leavitt, D.D., Starkschall, D. (eds.) *Proc. XII Int. Conf. on the Use of Computers in Radiation Therapy*, vol. XII, pp. 16–18. Medical Physics Publishing, Salt Lake City, Madison (1997)
19. Andreo, P.: Monte Carlo techniques in medical radiation physics. *Physics in Medicine and Biology* 36(7), 861–920 (1991)
20. Ma, C.M., Jiang, S.B.: Monte Carlo modelling of electron beams from medical accelerators. *Physics in Medicine and Biology* 44(12), R157–R189 (1999)
21. Rogers, D.W.O.: Fifty years of Monte Carlo simulations for medical physics. *Physics in Medicine and Biology* 51(13), R287–R301 (2006)
22. Krieger, T., Sauer, O.A.: Monte Carlo- versus pencil-beam-/collapsed-cone-dose calculation in a heterogeneous multi-layer phantom. *Physics in Medicine and Biology* 50(5), 859–868 (2005)
23. Ahnesjo, A., Saxner, M., Trepp, A.: A pencil beam model for photon dose calculation. *Medical Physics* 19(2), 263–273 (1992)
24. Ahnesjo, A.: Collapsed cone convolution of radiant energy for photon dose calculation in heterogeneous media. *Medical Physics* 16(4), 577–592 (1989)

25. Bayucan, A., Henderson, R.L., Lesiak, C., Mann, B., Proett, T., Tweten, D.: Portable Batch System: External reference specification. Technical report, MRJ Technology Solutions, 2672 Bayshore Parkway, Suite 810, Mountain View, CA 94043 (July 2005)
26. Henderson, R.L.: Job scheduling under the portable batch system. In: IPPS '95: Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing, pp. 279–294. Springer, London (1995)
27. Jackson, D., Snell, Q., Clement, M.: Core algorithms of the Maui scheduler. In: Feitelson, D.G., Rudolph, L. (eds.) JSSPP 2001. LNCS, vol. 2221, pp. 87–102. Springer, Heidelberg (2001)
28. Pinchak, C., Lu, P., Goldenberg, M.: Practical heterogeneous placeholder scheduling in overlay metacomputers: Early experiences. In: Feitelson, D.G., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2002. LNCS, vol. 2537, pp. 205–228. Springer, Heidelberg (2002)
29. Goldenberg, M., Lu, P., Schaeffer, J.: TrellisDAG: A system for structured DAG scheduling. In: Feitelson, D.G., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2003. LNCS, vol. 2862, pp. 21–43. Springer, Heidelberg (2003)
30. Aida, K., Natsume, W., Futakata, Y.: Distributed computing with hierarchical master-worker paradigm for parallel branch and bound algorithm. In: Proceedings of the Third IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'03), pp. 156–163 (May 2003)
31. Iverson, M.A., Özgüner, F.: Hierarchical, competitive scheduling of multiple DAGs in a dynamic heterogeneous environment. *Distributed Systems Engineering* 6, 112–120 (1999)
32. Milojicic, D.S., et al.: Peer-to-peer computing. Technical Report HPL-2002-57, HP Laboratories Palo Alto (March 2002)
33. von Laszewski, G., Hategan, M.: Workflow concepts of the Java CoG Kit. *Journal of Grid Computing* 3(3-4), 239–258 (2006)
34. Foster, I.: Globus toolkit version 4: Software for service-oriented systems. In: IFIP International Conference on Network and Parallel Computing, pp. 2–13. Springer, Heidelberg (2005)
35. Ylonen, T.: The Secure Shell (SSH) protocol architecture (January 06) Status: Proposed Standard
36. Livny, M., Basney, J., Raman, R., Tannenbaum, T.: Mechanisms for high throughput computing. *SPEEDUP Journal* 11(1) (June 1997)
37. Basney, J., Livny, M.: Deploying a high throughput computing cluster. In: Buyya, R. (ed.) *High Performance Cluster Computing: Architectures and Systems*, vol. 1, Prentice Hall PTR, Englewood Cliffs (1999)
38. Raman, R., Livny, M., Solomon, M.: Resource management through multilateral matchmaking. In: Proceedings of the Ninth IEEE Symposium on High Performance Distributed Computing (HPDC9), Pittsburgh, PA, pp. 290–291. IEEE Computer Society Press, Los Alamitos (2000)
39. Tannenbaum, T., Wright, D., Miller, K., Livny, M.: Condor – a distributed job scheduler. In: Sterling, T. (ed.) *Beowulf Cluster Computing with Linux*, MIT Press, Cambridge (2001)
40. Blythe, J., Jain, S., Deelman, E., Gil, Y., Vahi, K., Mandal, A., Kennedy, K.: Task scheduling strategies for workflow-based applications in grids. In: Proceedings of CCGrid 2005. Cardiff, Wales, vol. 2, pp. 759–767 (May 2005)