A Grid Workflow-Based Monte Carlo Simulation Environment

Yaohang Li[1], Michael Mascagni[2], Robert van Engelen[3], Qin Cai[4]
[1]*Department of Computer Science*
North Carolina A&T State University
Greensboro, NC 27411, USA
[2,3]*Department of Computer Science and School of Computational Science and Information Technology*
Florida State University
Tallahassee, FL 32306-4530, USA
[4]*Department of Computer Science*
Indiana University
Bloomington, IN 47405, USA

Abstract

Monte Carlo methods provide enormous scope for realistic statistical modeling and simulation. The implementation of large-scale Monte Carlo applications on the grid benefits from state-of-the-art approaches to accessing resources in a computational grid. Workflow techniques allow one to describe and enact his simulation processes in a structured, manageable, and verifiable way. We developed the Grid-Computing Infrastructure for Monte Carlo Applications (GCIMCA) based on the Globus toolkit and the SPRNG library. The Globus toolkit facilitates the creation and utilization of a computational grid for large distributed computational jobs and the Scalable Parallel Random Number Generators (SPRNG) library is designed to generate practically infinite number of random number streams with favorable statistical properties for parallel and distributed Monte Carlo applications. GCIMCA provides grid services specific to grid-based Monte Carlo simulation applications, including the Monte Carlo subtask schedule service using the N-out-of-M strategy, the facilities of application-level checkpointing, the partial result validation service, and the intermediate value validation service. By taking advantage of emerging grid workflow paradigms and the facilities of GCIMCA, we implemented a Grid Workflow-based Monte Carlo (GWMC) simulation environment. Workflow management services are implemented to manage the Monte Carlo simulation process. Based on these services, we intend to provide a trustworthy and manageable grid-computing environment for large-scale and high-performance distributed Monte Carlo simulation applications.

1. Introduction

Monte Carlo methods provide solutions to a variety of mathematical problems through statistical sampling. They are important techniques for performing simulation and optimization in numerous fields of science and engineering, including nuclear physics, chemistry, meteorology, biology, and medicine. The applications using Monte Carlo methods are widely perceived as computationally intensive but naturally parallel. With more ambitious calculations by estimating more random samples, a Monte Carlo application is capable of reducing the statistical errors to any desired level (Srinivasan et al., 1999). By computing and analyzing random samples independently, Monte Carlo applications can be programmed in a dynamic *bag-of-work* model and fit into the master-worker paradigm. In a parallel environment using the master-worker paradigm, the master partitions the task, schedules subtasks to workers, and receives results when the workers complete their assigned work (Basney et al., 1999). The subsequent growth of computer power, especially that of the parallel/distributed computing systems and the newly emerging grid computing systems, has made large-scale distributed Monte Carlo computation possible and practically effective.

Large-scale Monte Carlo computation consumes large amounts of computational power, and depends on high-quality parallel random number generators with good quality. On the one hand, grid computing is characterized by large-scale sharing and cooperation of dynamically distributed resources, such as CPU cycles, communication bandwidth, and data to constitute a computational environment (Foster et al., 2001). A computational grid based on grid-computing techniques can, in principle, provide a tremendously large amount of CPU cycles to a Monte Carlo application. The Globus software toolkit (Foster & Kesselman, 1997) provides software tools and services to build computational grid infrastructures for grid-based applications. On the other hand, the SPRNG (Scalable Parallel Random Number Generators) (Mascagni & Srinivasan, 2000) library is designed to use parameterized pseudorandom number generators to provide independent random number streams. The SPRNG library provides uniform programming interfaces for the Linear Congruential Generator (LCG), Prime Modulus Linear Congruential Generator (PMLCG), additive Lagged-Fibonacci Generator (LFG), Multiplicative Lagged-Fibonacci Generator (MLFG), and Combined Multiple Recursive Generator (CMRG). Some generators in the SPRNG library can provide up to $2^{78000} - 1$ independent random number streams (SPRNG Website, 2003) with sufficiently long period, which have favorable inter-stream and cross-stream properties in a statistical sense. These generators can meet the random number requirements of most distributed Monte Carlo applications. Furthermore, by analyzing the statistical nature of Monte Carlo applications and the cryptographic aspects of these underlying random number generators, our previous research (Li & Mascagni, 2002; Li & Mascagni, 2003; Li & Mascagni, 2003) developed techniques to improve the performance and trustworthiness of Monte Carlo computations on the grid. Recently, in (Li et al., 2003), we developed a grid middleware, a Grid-Computing Infrastructure for Monte Carlo Applications (GCIMCA) to assemble "all the pieces of the puzzle" for large-scale grid-based Monte Carlo analysis.

In addition to the issues of performance and trustworthiness (Li & Mascagni, 2002), a computational grid also presents a number of other challenges, including heterogeneity of grid equipment, crossing administrative domain cooperation, and dynamism of grid resources. With the functionalities of the organization of complex high-performance computational tasks within a grid-computing environment, workflow management is emerging as one of the most important grid services to address these challenges and implement e-science process automation. The WfMC (Workflow Management Coalition) standard developed the Workflow Process Definition Language (WPDL) (Allen, 2001) to specify general workflow scenarios. However, the WPDL is too generalized and sophisticated for grid computing (Bivens, 2001; Cao et al., 2003). More specifically, the Grid Workflow specification based on XML (Bivens, 2001) is documented and used in the ASCI (Accelerated Strategic Computing Initiative) grid infrastructure. In this paper, we will adopt these workflow techniques to implement grid-based Monte Carlo simulation process automation. Based on this Grid Workflow specification, we are going to elucidate the implementation of the Grid Workflow-based Monte Carlo (GWMC) simulation environment. The GWMC simulation environment integrates the services of GCIMCA to provide a trustworthy and easy-to-manage grid-computing environment for large-scale and high-performance distributed Monte Carlo simulation applications.

The remainder of this paper is organized as follows. We illustrate the system architecture and the working paradigm of the GWMC simulation environment in Sections 2 and 3, respectively. In Section 4, we discuss detailed implementations of the core services and facilities in GCIMCA, and the underlying middleware of GWMC simulation environment. Finally, Section 5 summarizes our conclusions and future research directions.

2. Architecture of GWMC Simulation Environment

The GWMC simulation environment is based on the grid middleware, GCIMCA (Li et al., 2003). GCIMCA is designed on the top of the grid services provided by Globus, (Globus Website, 2003) and supplies facilities and services for grid-based Monte Carlo applications. The Globus grid services include GRAM (Globus Resource Allocation Manager), GIS (Grid Information Service), GSI (Grid Security Infrastructure), and GridFTP. GRAM is used to implement Monte Carlo subtask remote-submission and manage the execution of each subtask. GIS provides information services, i.e., the discovery of the properties and configurations of grid nodes. GSI offers security services such as authentication, encryption and decryption for running Monte Carlo applications on the grid. GridFTP provides a uniform interface for data transport and access on the grid for GCIMCA. At the same time, the execution of each Monte Carlo subtask usually consumes a large amount of random numbers. SPRNG is the underlying pseudorandom number generator library in GCIMCA, providing independent pseudorandom number streams. As a grid middleware based on the grid services provided by Globus and the SPRNG library, GCIMCA provides higher-level services, including the *N-out-of-M* Monte Carlo subtask scheduling, application-level checkpointing, partial result validation, and intermediate value checking. Then, the GWMC grid workflow service

integrates the services of GCIMCA to provide workflow management services for grid-based Monte Carlo simulation applications. Figure 2.1 shows the architecture of the GWMC simulation environment.

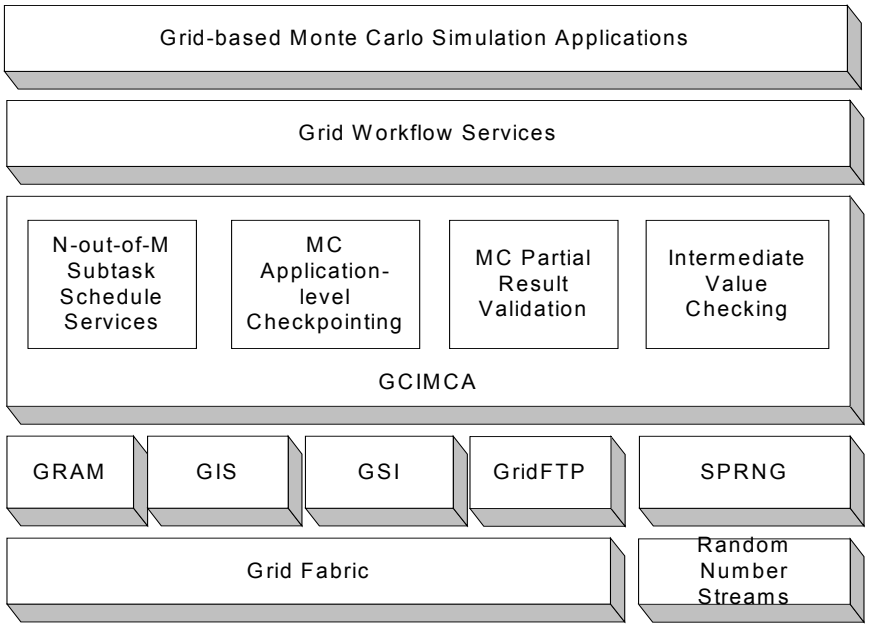

Figure 2.1 Architecture of GWMC Simulation Environment

3. Working Paradigm

3.1 Workflow Description of The Simulation Process

A workflow in the GWMC simulation environment can be decomposed into smaller units. These units are described as follows:

- Operation: Operations are the smallest elements in a grid workflow. Each operation in a grid workflow corresponds to a computational subtask and is usually excutable on a grid node.
- Sub-workflow: A sub-workflow is a flow of closely related operations that is to be executed in a predefined order on the grid resources within a virtual organization. Each sub-workflow represents a specific activity in an organization. Sub-workflows may be executed in parallel.
- Main-Workflow: A main-workflow can be represented as a flow of several loosely coupled activity described in a Monte Carlo simulation process. Each activity consumes various grid resources and can be represented by a sub-workflow.

3.2 Grid-based Monte Carlo Simulation Process Overview

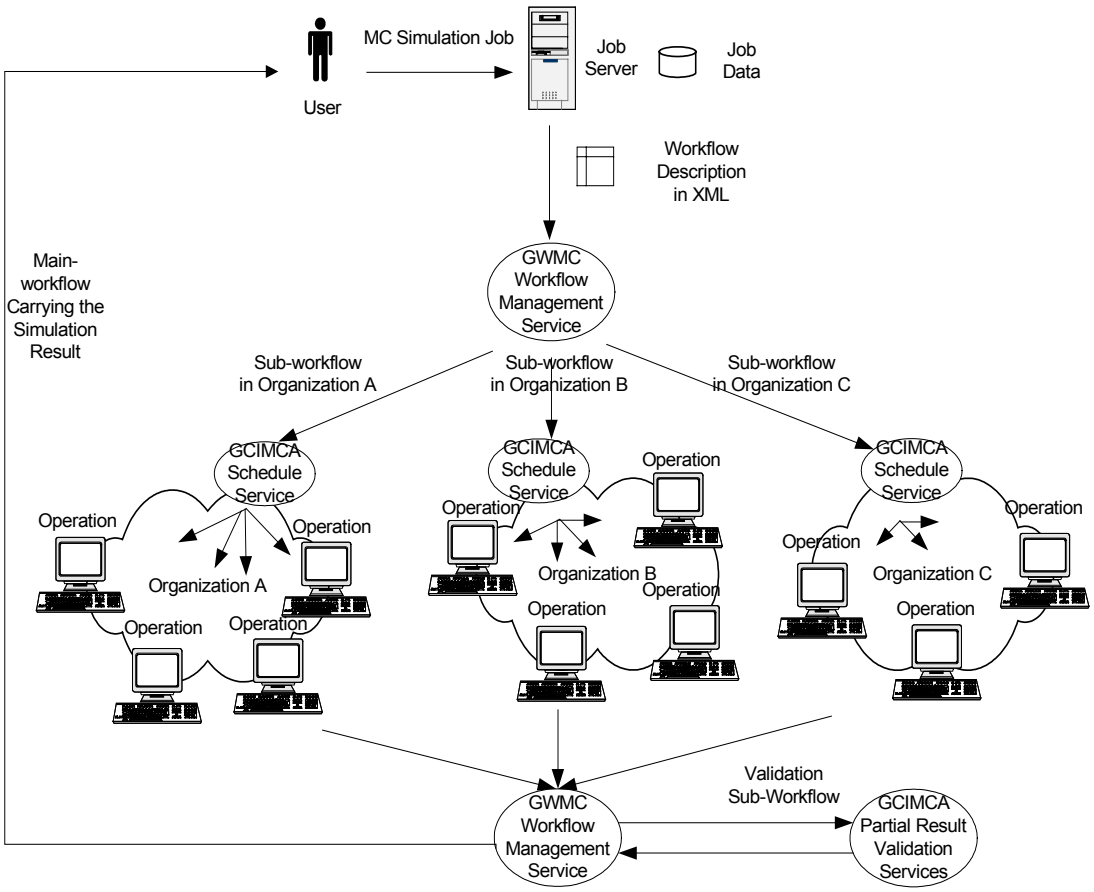

Figure 3.1 The Working Paradigm in the GWMC Simulation Environment

Figure 3.1 shows the working paradigm in GWMC simulation environment. A user submitting a Monte Carlo job description to the job server initiates the execution of a grid-based Monte Carlo application in the GWMC simulation environment. At the same time, the user prepares and stores the Monte Carlo job files, such as the executable binary and application data files, on the job server. Then, a grid workflow script to describe this Monte Carlo simulation process specified in XML is generated. According to the requirements of the main-workflow, the GWMC workflow management service dispatches the execution of sub-workflows to the GCIMCA schedule services. The GCIMCA schedule service manages the operations specified in its sub-workflow script. More specifically, in this case, each operation in the sub-workflow script actually refers to a Monte Carlo subtask. The GCIMCA schedule service then retrieves the Monte Carlo subtasks and is in charge of actually scheduling the subtasks. The subtasks are run on the grid resources in the organization, which is based on the master-worker model of a grid-based Monte Carlo application. When the subtasks are complete, the GWMC workflow management service then executes the partial result validation sub-workflow using the GCIMCA partial result verification services, which include partial result validation and intermediate value checking. After verification, the simulation results are sent back to the user, specifying the end of the execution of the Monte Carlo simulation workflow.

3.3 Job Submission

In GWMC simulation environment, a user provides different executable binary files, one for each system architecture on the grid. The remote compiler (Zhou, 2000) service is used to address this heterogeneity issue. A user can send source packages to a remote node of a specific system architecture with the remote compiler service running. Then, the remote compiler service compiles the source files, generates the executable files, and sends them back to the user. Using the remote compiler service, different executable codes for different platforms can be obtained.

Monte Carlo Job Description

```
JobName =          "Monte Carlo Integration"
JobDescription=
          "Execfile=http://sprng.cs.fsu.edu/mcint/mcintIntel.out
           Datafile=http://sprng.cs.fsu.edu/mcint/mcint.data
           Arg= -r
           Arch=INTEL
           Opsys=LINUX"
JobDescription=
          "Execfile=http://sprng.cs.fsu.edu/mcint/mcintSolaris.out
           Datafile=http://sprng.cs.fsu.edu/mcint/mcint.data
           Arg= -r
           Arch=SUN
           Opsys=Solaris26"
RequiredJobs =    20
MaxJobs =         40
ResultFileName =          mcintresult.dat
ResultLocation = http://sprng.cs.fsu.edu/mcint/result
Org=              cs.fsu.edu;csit.fsu.edu
Encryption=       YES
```

Figure 3.2 Sample of a Monte Carlo Job Description File for A Grid-based Monte Carlo Integration Application

To run a Monte Carlo application in the GWMC simulation environment, a user need to submit a Monte Carlo simulation job description file to the job server. The Monte Carlo job description file declares the information related to the Monte Carlo job, including the job name, locations of executable and data files, arguments, required hardware architectures and operating systems, number of subtasks, result file names and destinations, encryption option, and authenticated organization. Figure 3.2 shows a sample of a job description file for a grid-based Monte Carlo integration application. Based on the job description, the job server validates the Monte Carlo job, creates a Monte Carlo subtask pool, and creates a corresponding workflow file.

Figure 3.3 shows the main-workflow script corresponding to the Monte Carlo job description file in Figure 3.2. The data transfer directives at the beginning of the workflow script indicate the executable files and application data for the Monte Carlo integration job while those at the end specify the storage location of the result data. The data transfer directives instruct the workflow engine to execute the appropriate data

transfer program. The computation directive describes the computational process in the GWMC simulation environment.

```
<WorkFlow id = "main" name = "Monte Carlo Integration">
  <DataTransfer id = "MainData001">
    <Description> Executable Files </Description>
    <Argument name = "Execfile" value = "http://sprng.cs.fsu.edu/mcint/mcintIntel.out" />
    <Argument name = "Arch" value = "INTEL" />
    <Argument name = "Opsys" value = "LINUX" />
  </DataTransfer>
  <DataTransfer id = "MainData002">
    <Description> Executable Files </Description>
    <Argument name = "Execfile" value = "http://sprng.cs.fsu.edu/mcint/mcintSolaris.out" />
    <Argument name = "Arch" value = "SUN" />
    <Argument name = "Opsys" value = "Solaris26" />
  </DataTransfer>
  <DataTransfer id = "MainData002">
    <Description> Data Files </Description>
    <Argument name = "Datafile" value = "http://sprng.cs.fsu.edu/mcint/mcint.data" />
  </DataTransfer>

  <Computation>
    <ComputationAttribute name = "RequiredJobs" value = "20" />
    <ComputationAttribute name = "MaxJobs" value = "40" />
    <ComputationAttribute name = "Org" value = "cs.fsu.edu;csit.fsu.edu" />
    <ComputationAttribute name = "Encryption" value = "YES" />
  </Computation>

  <DataTransfer id = "MainResult">
    <Description> Data Files </Description>
    <Argument name = "Datafile" value = "http://sprng.cs.fsu.edu/mcint/result/mcintresult.dat" />
  </DataTransfer>
</WorkFlow>
```

Figure 3.3 The Main Workflow Script in XML Corresponding to the Monte Carlo Integration Job in Figure 3.2

According to the main workflow script, the corresponding sub-workflow scripts are then generated by the GWMC workflow management service. The operations in the sub-workflow specify the detail information of executing a Monte Carlo subtask. Figure 3.4 shows a sub-workflow script based on the main workflow script in Figure 3.3. In each computation directive, a Monte Carlo computational subtask is specified. Each subtask works on the same data but different random number streams. All these subtasks having the same order number indicate that they can be executed in parallel.

```
<SubWorkFlow id = "subworkflow1" name = "Monte Carlo Integration" organization = "cs.fsu.edu">
  <DataTransfer> … </DataTransfer>
  <Computation id = "mcintsubtask1" input = "mcintIntel.out" order = "1">
    <Argument name = "datafile" value = "mcint.data" />
    <ComputationAttribute name = "randomnumberstreamID" value = "1" type = "LCG"/>
    <ComputationAttribute name = "directory" value = "/etc/scratch/tmp" />
    <ComputationAttribute name = "checkpointfile" value = "/etc/scratch/subtask1.cpt" />
    <ComputationAttribute name = "partialresultfile" value = "/etc/scratch/result1.dat" />
    <ComputationAttribute name = "resultdirectory" value = "/etc/scratch/mcint/subtask1_result" />
  </Computation>
```

```
  <Computation id = “mcintsubtask2” input = “mcintIntel.out” order = “1”>
    <Argument name = “datafile” value = “mcint.data” />
    <ComputationAttribute name = “randomnumberstreamID” value = “2” type = “LCG”/>
    <ComputationAttribute name = “directory” value = “/etc/scratch/tmp” />
    <ComputationAttribute name = “checkpointfile” value = “/etc/scratch/subtask2.cpt” />
    <ComputationAttribute name = “partialresultfile” value = “/etc/scratch/result2.dat” />
    <ComputationAttribute name = “resultdirectory” value = “/etc/scratch/mcint/subtask2_result” />
  </Computation>
          …
  <DataTransfer> … </DataTransfer>
</SubWorkFlow>
```

Figure 3.4 A Sub-workflow Script

3.4 Passive-Mode Subtask Scheduling

Actually, in the GWMC simulation environment, the scheduling of Monte Carlo subtasks specified in the workflow script is completed by the GCIMCA schedule service. Unlike the design of most existing distributed and parallel computing systems, such as Condor (Litzkow et al., 1998), Javelin (Christiansen et al., 1997), Charlotte (Baratloo et al., 1996), and HARNESS (Beck et al., 1999), which use an active scheduling mode to dispatch subtasks, the GCIMCA schedule service uses a passive scheduling mode. In an active scheduling mode, the schedule service needs to keep checking the status of computational nodes to schedule tasks to the capable ones. Also, the schedule service must keep track of each running subtask. In contrast, using the passive scheduling mode in GCIMCA, a schedule service provider sends applications to the job server to apply for a subtask only when it has computational nodes available within its organization and ready for work. The management responsibility for the execution of each subtask is decentralized to the GCIMCA schedule service providers. The advantage of using the passive scheduling mode here is to reduce the workload, or more specifically, the requirements of network connection bandwidth of the job server. In GCIMCA, most of the communication load is between a GCIMCA schedule service provider and the computational nodes within the organization usually having connection via a high-speed LAN. On the other hand, the communication between the job server and the GCIMCA schedule service providers, which is usually through a WAN with relatively low bandwidth, is minimized.

The job server manages the jobs submitted from the users in the GWMC simulation environment, and processes subtask applications from the GCIMCA schedule service providers. It is the GCIMCA schedule service provider that retrieves the information related to a subtask, forms the subtask described in Globus RSL (Resource Specification Language) (Globus Website, 2003), and actually schedules the subtask to a grid node. The job management functionalities of GRAM are utilized to run subtasks on a remote grid node. Figure 3.5 illustrates the GCIMCA implementation for remotely executing a Monte Carlo subtask based on GRAM. When a Monte Carlo subtask is scheduled on a grid node, a process running the GCIMCA subtask callback function is created so as to listen to the status as it changes on the running subtask. Depending on the status of the

running subtask, the callback function takes corresponding actions, such as reporting errors, submitting partial result files, or rescheduling the subtask with checkpoint data.

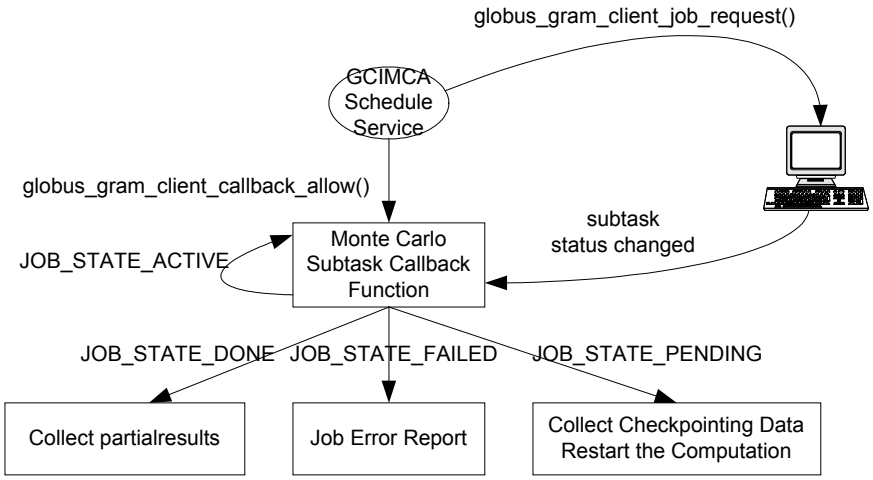

Figure 3.5 Remote Execution of a Monte Carlo Subtask

The operations of the scheduling a Monte Carlo subtask is described using the "RestartLoop" directive proposed in (Bivens, 2001) in a workflow script. Figure 3.6 shows such a workflow script.

```
<RestartLoop id = "MonteCarloSubtask">
  <Monitor id = "restart-subtask" activityToMonitor = "MCSubtask">
    <Condition pattern = "JOB_STATE_DONE" continue = "no" exitVal = "DONE" />
    <Condition pattern = "JOB_STATE_FAILED" continue = "no" exitVal = "FAILED" />
    <Condition pattern = "JOB_STATE_PENDING" continue = "yes" exitVal = "CONTINUE" />
  </Monitor>

  <Computation id = "MCSubtask" input = "mcsubtask.out">
    <Argument name = "checkpoint" value = "checkpoint.dat" />
  </Computation>
</RestartLoop>
```

Figure 3.5 A Workflow Script of the Execution of a Monte Carlo Subtask in GWMC Simulation Environment

4. Implementation of GCIMCA Services

The GWMC simulation environment is based on the grid middleware, GCIMCA. Taking advantage of the fundamental grid services provided by the Globus toolkit and parallel random number streams by the SPRNG library, GCIMCA provides higher-level services, including the *N-out-of-M* Monte Carlo subtask scheduling, application-level checkpointing, partial result validation, and intermediate value checking.

4.1 *N-out-of-M* Scheduling Strategy

The main idea of the *N-out-of-M* strategy (Li & Mascagni, 2002; Li & Mascagni, 2003) for grid-based Monte Carlo computations is to schedule more subtasks than are required to tolerate possible delayed or halted subtasks on the grid to achieve optimal performance. The statistical nature of Monte Carlo applications allows us to enlarge the actual size of the computation by increasing the number of subtasks from N to M, where $M > N$. Each of these M subtasks uses its unique independent random number stream, and we submit M instead of N subtasks to the grid system. When N partial results are ready, we consider the whole task for the grid system to be completed. More theoretical analysis of the *N-out-of-M* strategy can be found in (Li & Mascagni, 2003).

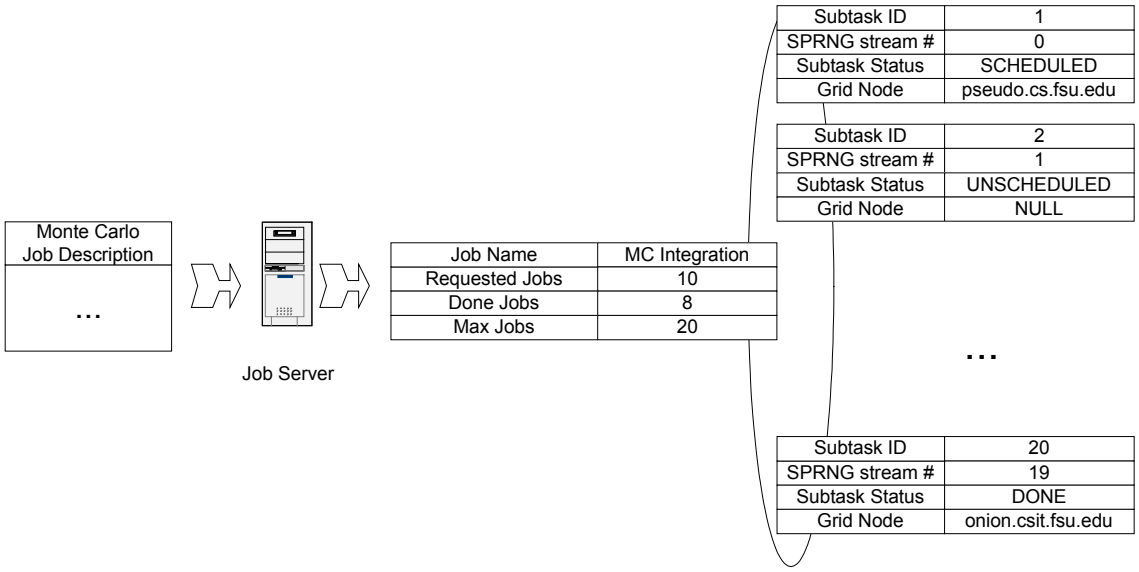

Figure 4.1 Implementation of the *N-out-of-M* Scheduling Strategy in GCIMCA

Figure 4.1 shows the implementation of the *N-out-of-M* scheduling strategy in GCIMCA. The Monte Carlo job description file from the user states the maximum number (M) of subtasks to be scheduled and the required number (N) of those to achieve a certain predetermined accuracy. Based on this, the job server sets up a subtask pool with the number of entries as M. Each entry of the pool describes the status of a subtask, including the subtask schedule status, random stream ID for the SPRNG library, the grid node if scheduled, and other implementation dependent details. The job server also maintains the statistics of completed subtasks. Once the number of completed subtasks reaches the number of requested subtasks, the job server will regard this Monte Carlo job as complete. A subtask-canceling signal will be sent to the GCIMCA schedule service providers that still have subtasks running related to this job.

Figure 4.2 shows the simulation of the *N-out-of-M* scheduling strategy on a computational grid comprised of nodes with various service rates. *10-out-of-10*, *10-out-of-20*, and *10-out-of-50* scheduling are compared in this simulation experiment. From the

Figure 4.2, we notice that we gain significant improvement in task completion time with a properly chosen value of *M*.

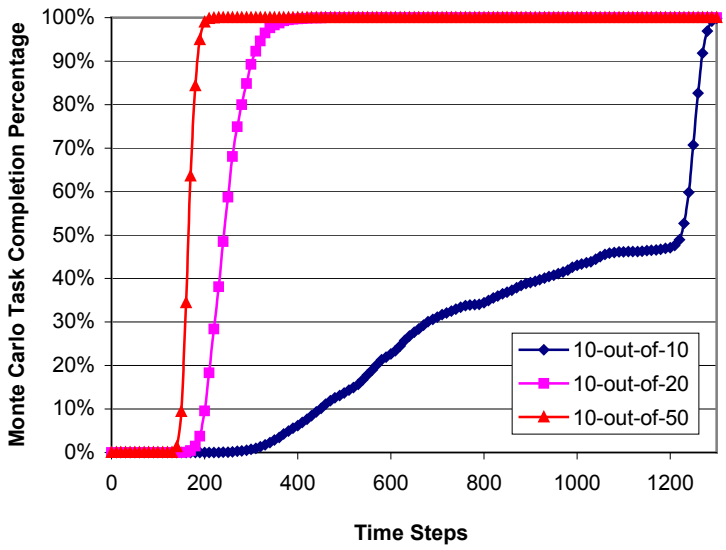

Figure 4.2 Simulations of the *N-out-of-M* Strategy on a Grid System

4.2 Monte Carlo Lightweight Checkpointing

A long-running computational task on a grid node must be prepared for node unavailability. Compared to process-level checkpointing (Litzkow et al., 1998), application-level checkpointing is much smaller in size and thus less costly. More importantly, the application-level checkpointing data is usually readily portable and is easy to migrate from one platform to another. Monte Carlo applications have a structure highly amenable to application-level checkpointing. Typically, a Monte Carlo application can be programmed in a structure that starts in an initial configuration, evaluates a random sample or a random trajectory, estimates a result, accumulates means and variances with previous results, and repeats this process until some termination conditions are met.

Thus, to recover an interrupted computation, a Monte Carlo subtask needs to save only a relatively small amount of information, which includes the current results based on the estimates obtained so far, the current status and parameters of the random number generators, and other relevant program information like the current iteration number. GCIMCA uses the *pack_sprng()* and *unpack_sprng()* functions (SPRNG Website, 2003) in the SPRNG library to store and recover the states of random number streams, respectively. At the same time, GCIMCA requires the Monte Carlo application programmer to specify the other checkpoint data, and also the location of the main loop to generate the checkpointing and recovery subroutines. Figure 4.3 shows the flowchart of GCIMCA's implementation of Monte Carlo application-level checkpointing and recovery.

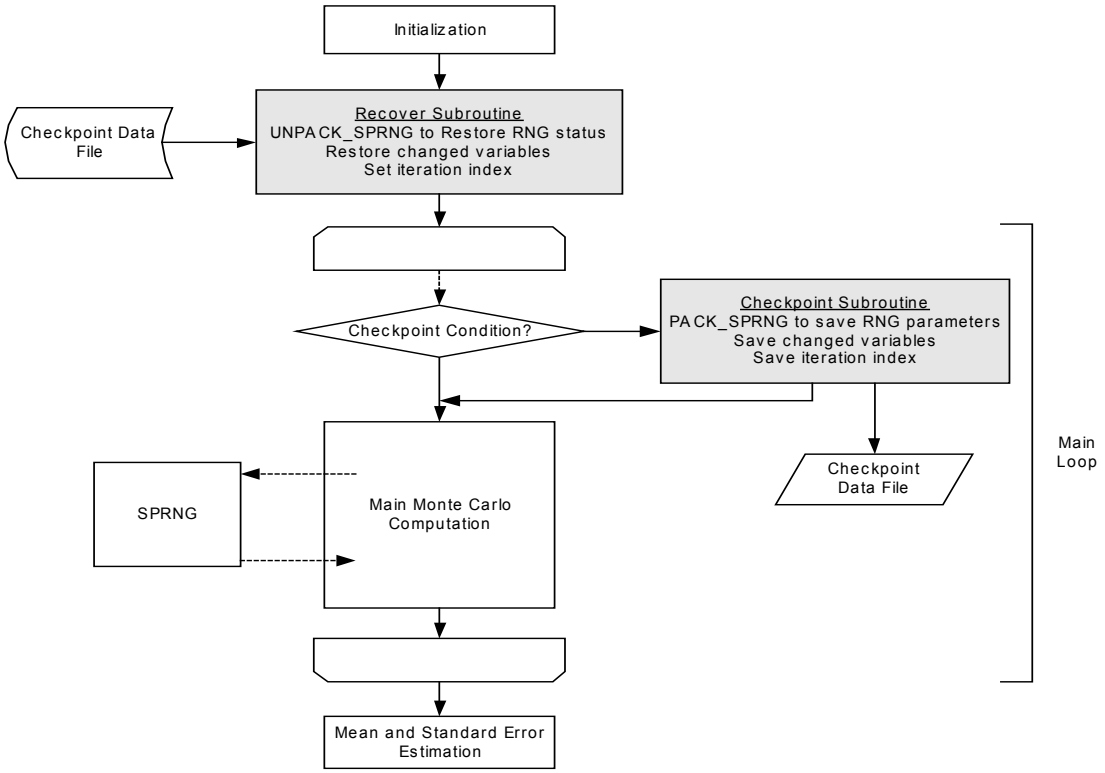

Figure 4.3 The GCIMCA Implementation of Monte Carlo Application-Level Checkpointing

4.3 Partial Result Validation and Intermediate Value Checking

Grid-based Monte Carlo applications are very sensitive to each partial result generated from subtasks running on widely distributed grid nodes. An erroneous computation of a subtask will most likely lead to the corruption of the whole grid Monte Carlo computation. To enforce the correctness and accuracy of grid-based Monte Carlo computations, GCIMCA provides a partial result validation service and an intermediate value checking service.

The partial result validation service takes advantage of the statistical nature of distributed Monte Carlo applications. In distributed Monte Carlo applications, we anticipate that the partial results will be approximately normally distributed. Based on all the partial results and a desired confidence level, the normal confidence interval is created. Then, each partial result is examined. If it is in the normal confidence interval, this partial result is considered as trustworthy; otherwise it is very suspicious. Discussion of the grid-based Monte Carlo partial result validation can be found in (Li & Mascagni, 2002; Li & Mascagni, 2003). To utilize the partial result validation service, GCIMCA requires the user to specify quantities in the partial result data files that are anticipated to conform to the approximately normal distribution. Then, when the Monte Carlo job is done, all these value files will be collected to compute the normal confidence interval and check each partial result. If a partial result is found suspicious, the particular subtask that

produced this partial result will be rescheduled on another grid node to perform further validation.

The intermediate value checking service is used to check if the assigned subtask from a grid node is faithfully carried out and accurately executed. The intermediate values are quantities generated within the execution of the subtask. To the node that runs the subtask, these values will be unknown until the subtask is actually executed and reaches a specific point in the program. On the other hand, to the owner of the application, certain intermediate values are either pre-known or very easy to generate. By comparing the intermediate values and the pre-known values, we can control whether the subtask is actually faithfully executed. The underlying pseudorandom numbers in the Monte Carlo applications are the perfect candidates to use as the intermediate values (Li & Mascagni, 2003). The intermediate value checking service in GCIMCA uses a simple strategy to validate a result from subtasks by tracing certain predetermined random numbers in the grid-based Monte Carlo applications. To utilize the intermediate value checking service, GCIMCA also requires user-level (programmer-level) cooperation. The application programmers need to save the value of the current pseudorandom number after every N pseudorandom numbers are generated. Thus, a record of the Nth, $2N$th, …, kNth random numbers used in the subtask are produced. When a subtask is complete, the verification service obtains this record and then re-computes the Nth, $2N$th, …, kNth random numbers applying the specific generator in the SPRNG library with the same seed and parameters as used in this subtask. A mismatch indicates problems during the execution of the subtask.

```
<SubWorkFlow id = “intvalchk1” name = “ResultVerification” organization = “cs.fsu.edu”>
  <DataTransfer id = “resultdata001”>
    <Argument name = “presult1” value = “result1.dat” host = “sprng.cs.fsu.edu”/>
  </DataTransfer>
  <Computation id = “intermediatevalchk1” input = “intermediatevalchk.out” order = “2”>
    <ComputationAttribute name = “randomnumberstreamID” value = “1” type = “LCG”/>
    <ComputationAttribute name = “validationresult” value = “vresult1.dat”/>
  </Computation>

  <DataTransfer id = “resultdata002”>
    <Argument name = “presult2” value = “result2.dat” host = “sprng.cs.fsu.edu”/>
  </DataTransfer>
  <Computation id = “intermediatevalchk2” input = “intermediatevalchk.out” order = “2”>
    <ComputationAttribute name = “randomnumberstreamID” value = “2” type = “LCG”/>
    <ComputationAttribute name = “validationresult” value = “vresult2.dat”/>
  </Computation>
  <DataTransfer> … </DataTransfer>
            …

<DataTransfer> … </DataTransfer>
  <Computation id = “presultvalidation” input = “presultvalidation.out” order = “3”>
    <ComputationAttribute name = “validationresult” value = “vresult.dat”/>
  </Computation>
  <DataTransfer> … </DataTransfer>
</SubWorkFlow>
```

Figure 4.4 A Sub-Workflow Script of Partial Result Verification

Figure 4.4 shows the sub-workflow script of partial result verification. The first two computation directives specify the intermediate value checking for each partial results. The same value of "order" attribute indicates that the intermediate value checking can be executed in parallel. The last computation directive specifies the partial result validation operation.

5. Conclusions

In this paper, we discussed utilizing the Globus software toolkit and the SPRNG library to implement a grid middleware of GCIMCA -- a grid-computing infrastructure for Monte Carlo applications. The *N-out-of-M* subtask schedule service, application-level checkpointing service, Monte Carlo partial result validation service, and intermediate value checking service are implemented in GCIMCA. Based on the facilities of GCIMCA, we developed the GWMC simulation environment to provide a workflow management service for grid-based Monte Carlo simulation processes for trustworthy and high-performance large-scale Monte Carlo computation. One of our computational biology applications, the grid-based nonequilibrium multiple-time scale molecular dynamics/Brownian dynamics simulations of ligand-receptor interactions in structured protein systems (Li et al., 2003), has demonstrated the power of large-scale grid-based Monte Carlo computation.

One of the drawbacks of the GWMC simulation environment design is that the end user has limited access to the workflow. Job submission is based on the job description file provided by the user and the workflow scripts are generated by the workflow management services. In the future, we plan to implement a graphic user portal to facilitate the composition of simulation workflow elements by the end user. Also, we plan to adopt the emerging OGSA (Open Grid Services Architecture) (Foster et al., 2003) into GCIMCA so that we can integrate the standard grid-computing services into grid-based Monte Carlo applications. Also, we plan to implement the remote checkpointing facilities using gSOAP (van Engelen & Gallivan, 2002) for Monte Carlo application-level checkpointing in a grid-computing environment. At the same time, we will also try to apply more real-life Monte Carlo applications on GCIMCA. A final goal is to experiment with the Monte Carlo-based services on non-Monte Carlo applications. The goal here being to study the extent to which these application-specific services can enhance other non-Monte Carlo grid computations.

References

1. Srinivasan, A., Ceperley, D. M. & Mascagni M., (1999). Random Number Generators for Parallel Applications. Monte Carlo Methods in Chemical Physics, v.105, pp. 13-36.
2. Basney, J., Raman, R., & Livny, M., (1999). High Throughput Monte Carlo. Proceedings of 9th SIAM Conference on Parallel Processing for Scientific Computing, San Antonio.

3. Foster, I., Kesselman, C., & Tueske, S., (2001). The Anatomy of the Grid. International Journal of Supercomputing Applications, v.15(3).
4. Foster, I., & Kesselman, C., (1997) Globus: A Metacomputing Infrastructure Toolkit. International Journal of Supercomputer Applications, v.11(2), pp. 115-128.
5. Mascagni, M., & Srinivasan, A., (2000). Algorithm 806: SPRNG: A Scalable Library for Pseudorandom Number Generation. ACM Transactions on Mathematical Software, v. 26, pp. 436-461.
6. SPRNG website, http://sprng.cs.fsu.edu.
7. Li, Y., & Mascagni, M., (2002). Grid-based Monte Carlo Application. Lecture Notes in Computer Science, GRID2002, v. 2536, pp. 13-25.
8. Li, Y., & Mascagni, M. (2003). Analysis of Large-scale Grid-based Monte Carlo Applications. Special issue of International Journal of High Performance Computation Applications.
9. Li, Y., & Mascagni, M. (2003). Improving Performance via Computational Replication on a Large-Scale Computational Grid. Proceedings of the GP2PC at the IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID2003, Tokyo.
10. Globus website, http://www.globus.org.
11. Litzkow, M., Livny, M., Mutka, M. (1988). Condor - A Hunter of Idle Workstations. Proceedings of 8th International Conference of Distributed Computing Systems, pp. 104-111.
12. Christiansen, O., Cappello, P., Ionescu, M. F., Neary, M. O., Schauser, K. E., & Wu, D. (1997). Javelin: Internet-Based Parallel Computing Using Java. Concurrency: Practice and Experience, v. 9(11), pp. 1139-1160.
13. Baratloo, A., Karaul, M., Kedem, Z., & Wyckoff, P., (1996). Charlotte: Metacomputing on the Web. Proceedings of 9th International Conference on Parallel and Distributed Computing Systems.
14. Beck, Dong, Fagg, Geist, Gray, Kohl, Miliardi, Moore, K., Moore, T., Papadopoulous, P., Scott, S., & Sunderam V., (1999). HARNESS: a next generation distributed virtual machine. Journal of Future Generation Computer Systems, v. 15.
15. Zhou, M., (2000). A Scientific Computing Tool for Parallel Monte Carlo in a Distributed Environment. Ph.D. Dissertation, University of Southern Mississippi.
16. Foster, I., Kesselman, C., Nick, J. M., & Tuecke, S., (2003). The Physiology of Grid: Open Grid Services Architecture for Distributed Systems Integration. Draft.
17. van Engelen, R., & Gallivan, K. A., (2002). The gSOAP Toolkit for Web Services and Peer-to-Peer Computing Networks. Proceedings of proceedings of the GP2PC at the IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGrid2002, Berlin.
18. Li, Y., Mascagni, M., & van Engelen, R., (2003). GCIMCA: A Globus and SPRNG Implementation of a Grid Computing Infrastructure for Monte Carlo Applications. Proceeding of the International Multiconference in Computer Science and Computer Engineering, PDPTA'03.
19. Allen, R., (2001). Workflow: An Introduction. Workflow Handbook 2001, Workflow Management Coalition.

20. Bivens, H. P., (2001). Grid Workflow. Grid Computing Environments Working Group, Global Grid Forum.
21. Cao, J., Jarvis, S. A., Saini, S., & Nudd, G. R., (2003). GridFlow: Workflow Management for Grid Computing. Proceedings of 3rd International Symposium on Cluster Computing and the Grid, CCGRID2003, Tokyo.
22. Li, Y., Mascagni, M., & Peters, M. H., (2003). Grid-based Nonequilibrium Multiple-Time Scale Molecular Dynamics/Brownian Dynamics Simulations of Ligand-Receptor Interactions in Structured Protein Systems. Proceeding of the First International Workshop on Biomedical Computations on the Grid (BioGrid'03), Tokyo.