Simulation of Parallel and Distributed Systems: A Taxonomy and Survey of Tools

Anthony Sulistio, Chee Shin Yeo, and Rajkumar Buyya

Grid Computing and Distributed Systems (GRIDS) Laboratory
Dept of Computer Science and Software Engineering
The University of Melbourne, Australia
{anthony, csyeo, raj}@cs.mu.oz.au

Abstract: Simulation has been applied successfully for modelling large complex systems and understanding their behaviour, especially in the area of parallel and distributed systems. As such, there are an increasing number of tools being designed to simulate parallel and distributed systems and designers may face difficulties in selecting the correct tool for simulating their applications. Therefore, this paper aims to provide a better understanding of the various tools available for simulation of parallel and distributed systems by conducting a survey of them.

1. Introduction

Simulation is widely used for modelling real world processes in many different application areas, including manufacturing [15, 16], construction [17, 18] and computer science [6, 9]. It provides the study of various issues, such as feasibility, behaviour and performance without building the actual system, thus saving time, cost and effort. There are an increasing number of tools designed for the simulation of parallel and distributed systems. These tools facilitate concurrent simulation for modelling actual operations, behaviours and performance of multiple systems or processes and the simultaneous event communications between them.

For instance, several tools [2, 3, 4, 5] are constructed for simulation of grid computing environments since it is difficult to have easy access to ready-built grid platforms to evaluate the concepts and algorithms involved. Moreover, it is almost impossible to evaluate their performance under different scenarios, given the varying number of resources whose availability changes with time and users with different requirements in a grid computing environment. Simulation tools support the creation of a *repeatable* and *controllable* environment for performance evaluation. This facilitates grid computing researchers, educators, and students to conduct effective research, teaching, and learning with ease.

Figure 1 provides a categorization of the various simulation tools for different application areas. For this paper, we will be focusing on simulation tools for parallel and distributed systems. The tools discussed are primarily tools that are currently under research, rather than tools that are developed and used commercially. Simulation tools can also be differentiated by their simulation models which are based on discrete events, continuous events or hybrid (having both discrete and continuous events) and design models which are based on software libraries or visual.

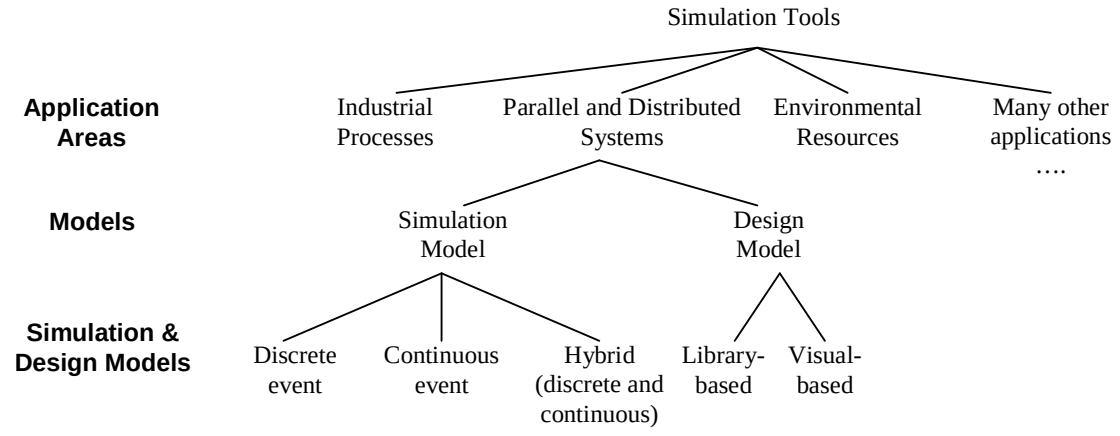

Figure 1: Categorization of simulation tools

The rest of the paper is organized as follows. Section 2 lists the different simulation tools and provides brief descriptions for some of them. A general comparison of the tools is then conducted in Section 3. The final section concludes the paper and specifies some further work.

2. Tools for simulation of Parallel and Distributed Systems

This section provides an overview of some the many available tools that simulate parallel and distributed systems and describes them in more details. Table 1 summarizes the tools.

No	Tool	Description	Developer	Target Applications
1.	SimJava	SimJava [1] provides process based discrete event simulation with animation facilities through a collection of entities communicating with each other. **http://www.dcs.ed.ac.uk/home/hase/simjava/**	University of Edinburgh, UK	Discrete event simulation
2.	Bricks	Bricks [2] is a performance evaluation tool that analyses and compares various scheduling schemes in high performance grid computing environment. **http://matsu-www.is.titech.ac.jp/~takefusa/bricks/**	Tokyo Institute of Technology, Japan	Grid simulation
3.	MicroGrid	MicroGrid [3] provides simulation tool that supports scalable simulation of grid applications using clustered resources in computational grid research. **http://www-csag.ucsd.edu/projects/grid/microgrid.html**	University of California at San Diego, USA	Grid simulation
4.	Simgrid	Simgrid [4] simulates distributed applications in heterogenous distributed environments. **http://grail.sdsc.edu/projects/simgrid/**	University of California at San Diego, USA	Grid simulation
5.	GridSim	GridSim [5] enables modelling and simulation of entities involved in parallel and distributed computing through the creation and monitoring of different resources. **http://www.buyya.com/gridsim/**	Monash University, Australia	Grid simulation
6.	SimOS	SimOS [6] provides a complete machine simulation environment that simulates computer hardware to model uniprocessor and multiprocessor computer systems. **http://simos.stanford.edu/**	Stanford University, USA	Computer system simulation
7.	Dimemas	Dimemas [7] is a performance analysis tool for message passing programs. It simulates the time behaviour of a parallel application on a machine or a cluster of machines modelled by a set of performance parameters. **http://www.cepba.upc.es/dimemas/**	The Technical University of Catalonia, Spain	Parallel program simulation
8.	SvPablo	SvPablo [8] is a performance analysis tool that captures and analyses data from serial and parallel programs. **http://www-pablo.cs.uiuc.edu/Software/Pablo/pablo.htm**	University of Illinois at Urbana-Champign, USA	Parallel program simulation
9.	MPISim	MPISim [9] is a parallel simulator for parallel programs using Message Passing Interface (MPI). **http://pcl.cs.ucla.edu/projects/mpisim/**	University of California at Los Angeles, USA	Parallel program simulation
10.	Ptolemy 2	Ptolemy 2 [10] is a Java-based library that supports heterogeneous, concurrent modelling and design. **http://ptolemy.eecs.berkeley.edu/ptolemyII/**	University of California at Berkeley, USA	Concurrent System simulation

No	Tool	Description	Developer	Target Applications
11.	DaSSF	DaSSF [11] is a C++ implementation of Scalable Simulation Framework (SSF) that enables scalable process-oriented and event-oriented simulation of large computer networks. **http://www.cs.dartmouth.edu/~jasonliu/projects/ssf/**	Darmouth College, USA	Computer network simulation
12.	Pamela	Pamela [12] is a process-oriented performance simulation language that models the interaction of the parallel program and the machine. **http://www.pds.twi.tudelft.nl/pamela/**	Delft University of Technology, Netherlands	Parallel program simulation
13.	Pepa	Pepa [13] known as Performance Evaluation Process Algebra is used for modelling concurrent systems which cooperate and share work tasks. **http://www.dcs.ed.ac.uk/pepa/**	University of Edinburgh, UK	Concurrent system simulation
14.	Clue	Clue [14] is a simulation and performance assessment tool for parallel programs using message passing libraries for communication. **http://www.math.tuwien.ac.at/~aurora/group5/node27.html**	Vienna University of Technology, Austria	Parallel program simulation

Table 1: A summary of the distributed simulation tools

2.1 SimJava

SimJava [1] is a Java-based toolkit that enables simulation of complex systems. It is built based on a discrete event simulation kernel and is able to provide visual animation of the simulated objects during simulation. As such, Java is used to ensure easy incorporation of these live diagrams into web pages and portability across platforms. SimJava aims to provide a set of foundation classes to enable easy creation and animation of discrete event simulation models. SimJava comprises of three packages as explained in Table 2.

Package	Purpose
eduni.simjava	This builds text only java simulation and produces a trace file as the output by default.
eduni.simanim	This provides the applet template for easy building of the visualization of a simulation and is tightly integrated with the text-only eduni.simjava.
eduni.simdiag	This is a collection of JavaBeans based classes for displaying simulation results.

Table 2: Packages of SimJava

Simjava is designed for simulating static networks of active entities which communicate by sending passive event objects via ports. It is able to support the simulation of complex models and internal communication by programming using the package. Therefore, SimJava can be applied to model hardware and distributed software systems such as communication protocols, parallel software modelling and computer architectures [1].

Package eduni.simjava

Building a simulation using simjava involves declaring the entities of the system and specifying their behaviour. These entities run in parallel and communicate with one another through the passing of events. Each entity is modelled by extending the Sim_entity class to include its behaviour. Sim_entity provides basic simulation methods as summarized in Table 3.

Method	Usage
sim_schedule	Sends events to other entities via ports.
sim_hold	Pauses for some simulation time.
sim_wait	Waits for an event to arrive.
sim_select	Selects events from the queue.
sim_trace	Writes a time stamped message to the trace file.

Table 3: Methods of Sim_entity

The running of the simulation is managed by Sim_system which maintains a timestamp ordered queue of all the events to be executed. The events are sorted by simulation timestamp to ensure that no events arrive out of order. When an entity performs a function (eg. pause for a time interval by calling sim_hold(1.23)), Sim_system will stop the entity and reschedule it by updating its next execution time in the queue as shown in Figure 2. When no entities are running, Sim_system will pop the next event from the queue and restart the next waiting entity. The simulation is completed when the queue contains no more events. Running the simulation produces a trace file which records all the events that occur and monitors the progress of each entity [1].

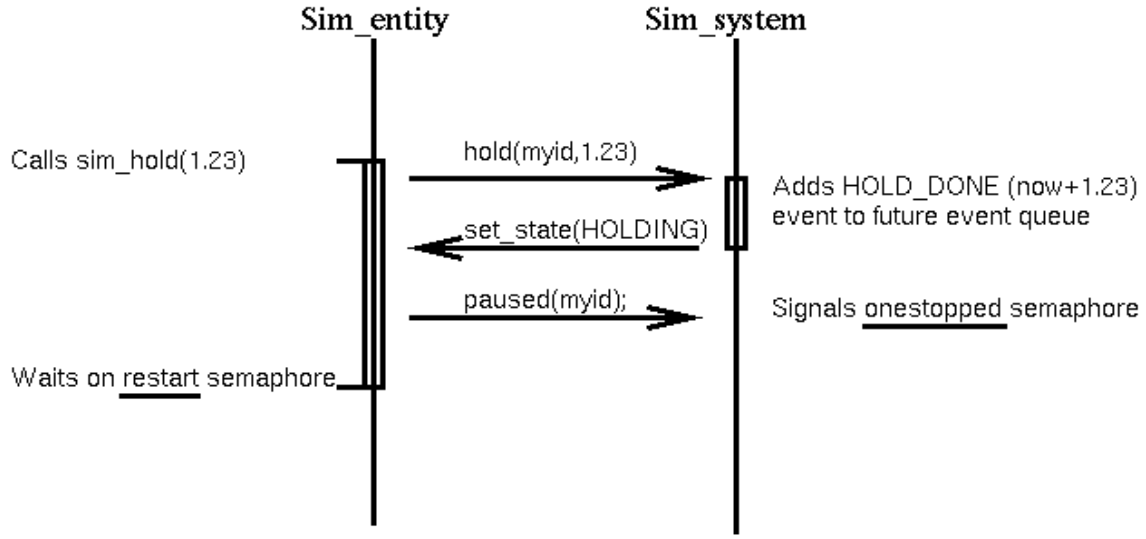

Figure 2: Interaction between Sim_entity and Sim_system in eduni.simjava

Package eduni.simanim

simanim supports visual representation and animation of a simulation model. To build an animated model, the user will build an applet that extends Anim_applet to customize the settings (using the method anim_init()) and layout the entities and the links between the entities. When the simulation is run, events are shown moving along the links and entities switch according to their states. An example is shown in Figure 3 where the solid/blue circle representing the event will move along the link and the entity switch between idle and busy states [1].

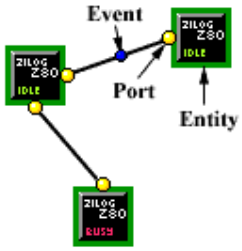

Figure 3: Animation in eduni.simanim

Package eduni.simdiag

simdiag provides Java Bean classes for displaying diagrams and graphs to summarize the results of the simulation. It supports two kinds of beans: The trace event listener which listens to the stream of events during the running of the simulation and the graph event listener which plots the graphs. Figure 4 shows the Sim_anim classes generating the events which is captured and displayed in a timing diagram and then stored to a tracefile by the trace saver. Figure 5 illustrates how each entity changes state over time [1].

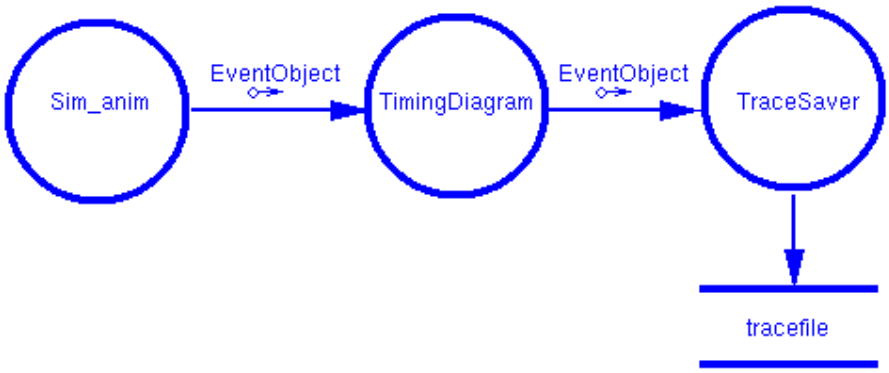

Figure 4: Beans Event Interface in eduni.simdiag

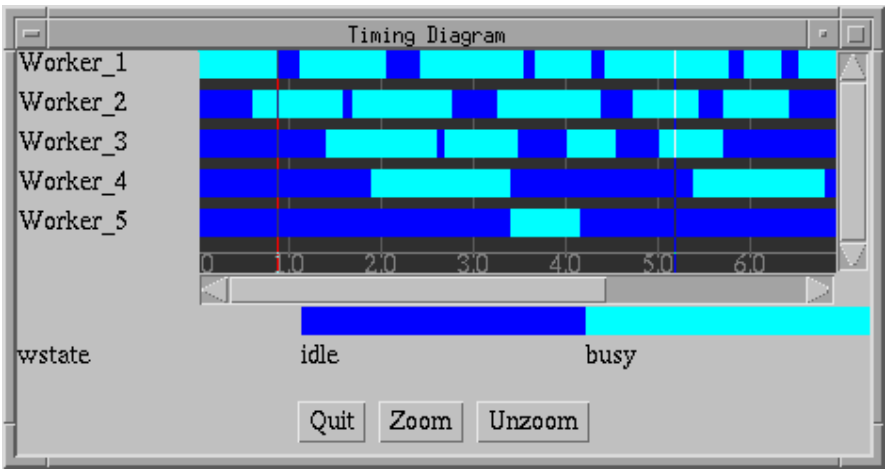

Figure 5: Timing diagram bean in eduni.simdiag

There are several projects that are built using SimJava, thus demonstrating its effectiveness in modelling simulations. These projects include [1]:

- GridSim (grid simulation toolkit) (Monash University, Australia)
- Distributed SimJava (MITRE Corp)
- Out of order simulation of SPARC-GP (ISTY Informatique, Versailles)
- Concurrent modelling and simulation in java, presentation (I C Legrand, CERN)
- Firewire SimJava simulation (M Mason, University of Birmingham, UK)
- SIMPROD (simjava based tools for production systems)
- BlueTooth simulation using simjava (Ching Law, MIT)
- self-stabilizing replication applets (F Gardner, TU Darmstadt)
- JobShop simulation (Haifa University, Israel)
- HiPeX (City University)
- Neko Project (Swiss Federal Institute of Technology, Lausanne)

2.2 Bricks

Bricks [2] is a performance evaluation system (written in java) that allows analysis and comparison of various scheduling schemes in a high performance global computing environment. Bricks can simulate various behaviours of global computing systems such as the behaviour of networks and resource scheduling algorithms.

Bricks provides simulation of various behaviours of resource scheduling algorithms, programming modules for scheduling, network topology of clients and servers in global computing systems and processing schemes for networks and servers. Bricks also gathers information on global computing resources for resource scheduling algorithms systematically to monitor and predict the resources in the global computing environment. It provides some components for monitoring, predicting and scheduling the jobs in the simulated network. This component-based architecture enables components to be replaced to simulate different system algorithms and allows incorporation of existing global computing components via its foreign interface [2].

Figure 6 shows the architecture of Bricks. Bricks consists of the Global Computing Environment and the Scheduling Unit which coordinates the simulation behaviour of global computing systems. As such, Bricks operates as a discrete event simulator of a queuing system in virtual time. Table 4 describes the modules in the Global Computing Environment which represents the global computing simulation environment, while Table 5 describes the modules in the Scheduling Unit that models a canonical scheduling framework for global computing systems [2].

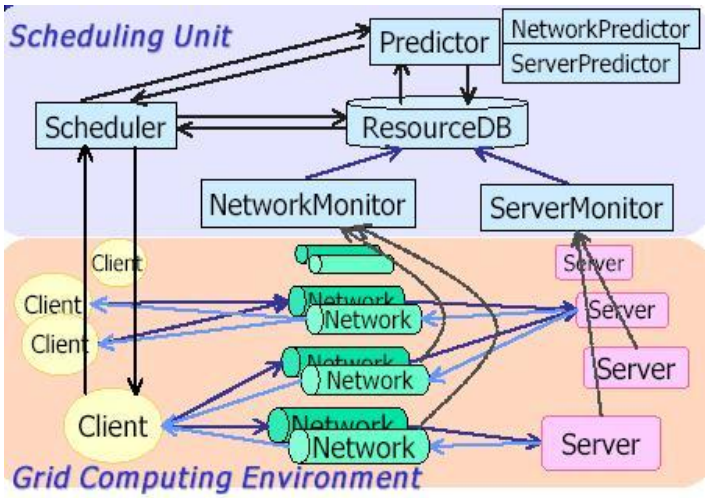

Figure 6: Architecture of Bricks

Module	Purpose
Client	Represents the user of the grid system who invokes grid computing jobs containing information such as number of executed instructions and amount of data transmitted.
Network	Represents the network interconnecting the Client and the Server. Both the network and server are represented using queues.
Server	Represents computational resources of the given global computing system.

Table 4: Modules of Global Computing Environment

Module	Purpose
NetworkMonitor	Measures network bandwidth and latency in global computing environments and stores the measured values in ResourceDB.
ServerMonitor	Measures performance, load and availability of server machines and stores the measured values in ResourceDB.
ResourceDB	Works as a scheduling-specific database, storing the various measurement values. These values are accessed by the Predictor and Scheduler to make forecasts and scheduling decisions.
Predictor	Extracts the measured resource information of certain time duration from ResourceDB and predicts availability of resources. The predicted information is often used for scheduling of a new global computing task.
Scheduler	Allocates a new task invoked by a client on suitable server machine(s) and makes scheduling decisions based on the resource information provided by ResourceDB and Predictor.

Table 5: Modules of Scheduling Unit

2.3 MicroGrid

The MicroGrid [3] tool provides a platform to develop and implement a virtual grid infrastructure to run a Globus application for the study of grid resource management issues involved. It allows systematic design and evaluation of middleware, applications and network services for the Computational Grid.

A virtual grid infrastructure will allow repeatable and controllable experimentations with dynamic resource management techniques, thus reducing effort for simulation and increasing observability of experiments. MicroGrid provides broad and full-scale experimentation using different control algorithms over a wide variety of Grid configurations [3].

MicroGrid aims to achieve the following goals [3]:

- Support for scalable simulation of Grid applications using a wide variety of scalable clustered resources,
- Support for realistic Grid software environment which enables the simulated applications to run with identical Application Program Interfaces (APIs) and thus behave accurately,
- Support for configuring Grid resource performance attributes, and
- An open software environment, enabling easy extensions and improvements to the capabilities of MicroGrid.

The functionality of the MicroGrid is to enable users to conduct experiments by simulating applications on a virtual Grid environment. It should be flexible enough to allow accurate experimentation on heterogeneous physical resources, as shown in Figure 7. This figure illustrates the MicroGrid exploring a range of virtual resources and

executing jobs on a range of physical resources. The MicroGrid implementation supports Grid applications which use the Globus Grid middleware infrastructure [3].

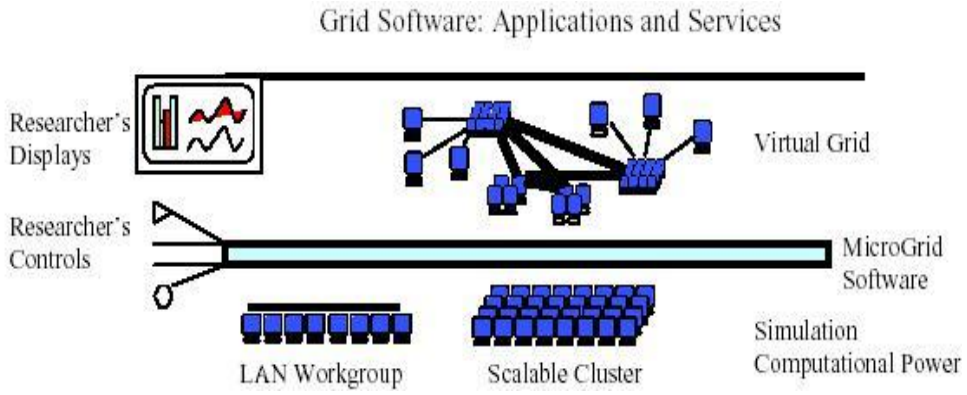

Figure 7: Architecture of MicroGrid

The MicroGrid simulation comprises of separate local resource simulations which provide a virtual grid resource environment and the network simulator which captures the interaction between these local resources. Each component in the architecture poses the challenge of constructing a highly accurate virtual Grid. These challenges are described in Table 6 [3].

Challenge	Description
Virtualization	Ensures that the application only perceives the virtual Grid resources (host names and networks) which are independent of the physical resources being utilized.
Global Coordination	Provides a consistent global simulation of different quantities of varying resources on heterogenous physical resources and ensure global coordination of simulation progress.
Resource Simulation	Each virtual resource must be modelled accurately as an element of overall simulation. Resource simulation can be of two types: • computing (eg. host, CPU, disk) • communication (eg. network)

Table 6: Challenges of MicroGrid simulation

2.4 Simgrid

Simgrid [4] is a toolkit that provides core functionalities for the evaluation of scheduling algorithms in distributed applications in heterogenous, computational Grid environment. Simgrid aims to be a tool that can be used to build domain-specific simulations to facilitate scheduling research with the following goals:

- Provides the right model and level of abstraction for its intended purpose
- Able to rapidly prototype and evaluate scheduling algorithms
- Enables more realistic simulation than previous research
- Generates correct and accurate simulation results

Simgrid provides a set of core abstractions and functionalities that helps to quickly build simulators for specific application domains and computing environment. Simgrid performs event-driven simulation that is based on the modelled resources. Therefore, it provides mechanisms to model performance characteristics either as constants or from traces. Traces enable the simulation of arbitrary performance changes as seen in real resources [4]. As such, the Simgrid toolkit includes features such as:

- Resource models for CPUs and network links

- Arbitrary, dynamic, trace-based resource performance metrics
- Resource time-sharing and time-slicing
- Task dependencies
- Performance prediction error simulation
- Flexible simulation stopping conditions
- Simple API

In the Simgrid approach, resources (such as CPUs and network links) are treated as unrelated resources, so no interconnection topology is imposed between the resources. This provides the flexibility for Simgrid to simulate a wide range of computing environments as the user will specify his topology requirements based on the application-specific requirements [4].

In addition, Simgrid does not differentiate between computations and data transfers. Both are seen as tasks and again, the user has to ensure that computations are scheduled on processors and file transfers on network links [4].

Figure 8 shows an example of an application with four computational tasks (C1, C2, C3, C4) and four data transfer tasks (T1, T2, T3, T4). On the left side is the traditional DAG (Directed Acyclic Graph) representation where edges symbolize both data transfers and task dependencies. On the right side is the simgrid model where edges symbolize task precedence and additional nodes are inserted for data transfers [4].

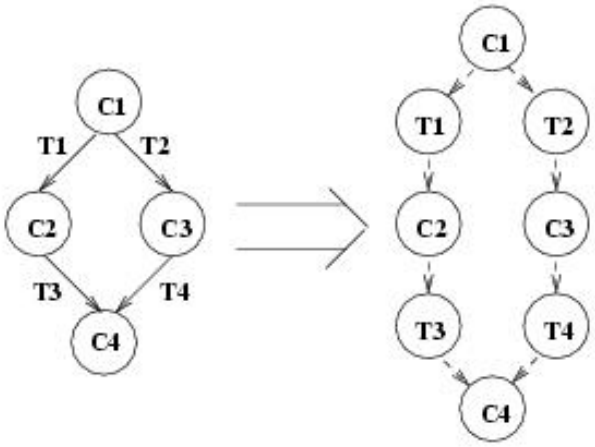

Figure 8: Task Model of Simgrid

Simgrid is implemented in C and provides a C API that allows the user to manipulate two data structures for resources and tasks respectively. A resource is described by a name, a set of performance related metrics and trace values, while a task is described by a name, a cost and a state [4].

There are few projects that are built on top of Simgrid [4]. These projects include:

- PSTSim (a simulator for the scheduling of Parameter Sweep applications)
- DAGSim (evaluating scheduling algorithms for DAG-structured applications in the presence of resource performance prediction errors).

2.5 GridSim

GridSim [5] is a java-based toolkit that allows simulation of different classes of heterogeneous resources, users, applications, resource brokers and schedulers in a distributed computing environment. This can then be used to simulate application schedulers for distributed computing systems such as clusters and grids.

GridSim is built upon a multi-layer architecture. The first layer is the portable and scalable Java's interface and runtime environment called Java Virtual Machine (JVM) whose implementation is available for single and multiprocessor systems including clusters. The second layer provides a basic discrete event infrastructure in SimJava [1] built on top of the interfaces provided by the first layer. The third layer is the GridSim toolkit that

provides the modelling and simulation of core Grid entities such as resources and information services using the discrete event services defined by the second layer. The fourth layer provides the simulation of resource aggregators called grid resource brokers or schedulers. The final layer focuses on application and resource modelling with different scenarios using services provided by the two lower-level layers for evaluating scheduling and resource management policies, heuristics, and algorithms [5].

Figure 9 shows a GridSim based simulation which contains the different entities as explained in Table 7.

Entity	Description
User	Represents a Grid user. Each user is different in terms of types of jobs created, scheduling optimization strategy, activity rate, time zone and deadline and budget affordability
Broker	Schedules the tasks for the assigned User based on the user's scheduling policy.
Resource	Represents a grid resource. Each resource is different in terms of number of processors, cost of processing, speed of processing, internal process scheduling policy, local load factor and time zone.
Grid information service	Provides resource registration services and monitors the list of resources available in the Grid.
Input and Output	Provides as the input and output links for the flow of information between the entities.

Table 7: GridSim entities

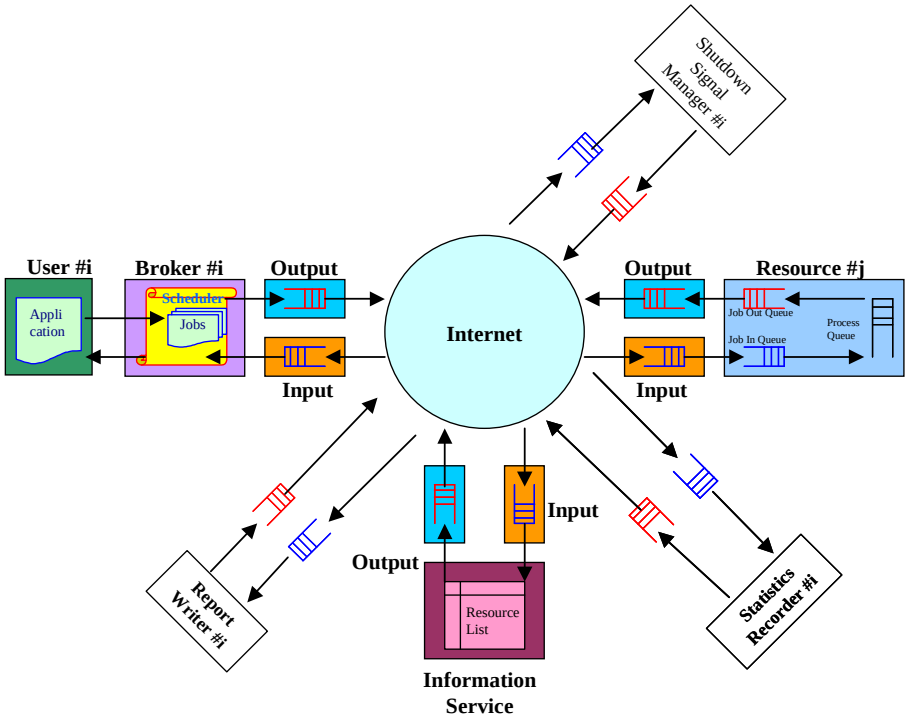

Figure 9: GridSim based simulations

The GridSim toolkit has been used in the simulation of:

- The Nimrod-G resource broker and economic scheduling algorithms for Grid computing
- The Libra cluster scheduler that uses co-operative computing economy for managing resources.

2.6 SimOS

SimOS [6] is a simulation environment that can model and simulate complete computer systems, including a full operating system and all application programs that run on it. This is achievable since SimOS provides an extremely fast simulation of system hardware and is able to control the level of simulation detail for modelling the full operating system.

SimOS has been used in the studying of different aspects of complex systems including computer architecture, operating systems and complex application workloads. It can be used to study the computer architecture such as the effects of new processor and memory system organizations on workloads such as large scientific applications and commercial database systems. It is designed to easily incorporate new hardware models to encourage investigation of new architectural designs. It can also be used to study the operating systems such as developing, debugging and performance tuning an SMP operating system for multiprocessors. This is well suited for studying shared memory parallel programs [6].

Figure 10 shows the SimOS environment. SimOS runs as a layer between the host machine and the target operating system. Host refers to the hardware and software on which SimOS runs, while target refers to the simulated hardware and the operating system and applications running on it. The figure shows that SimOS as a simulation layer that runs on top of general-purpose Unix multiprocessors such as SGI (Silicon Graphics Inc.) Challenge series. It simulates the hardware of an SGI machine in enough detail to support Irix version 5.2, the standard SGI version of Unix SVR4. Application workloads developed on SGI machines are run without any modifications on the simulated system. Each simulated hardware component in the SimOS layer has multiple implementations that vary in speed and detail. This clearly illustrates that SimOS is capable of running large and complex commercial applications available on the SGI platforms. SimOS provides various simulation modes for quick simulation as summarized in table 8 [6].

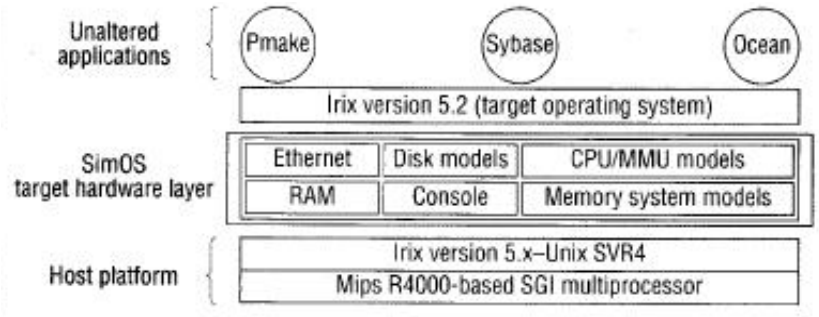

Figure 10: The SimOS environment

Mode	Description
Direct execution mode	Used when the host and target are similar to exploit the similarities between host and target by directly using the underlying machine's hardware to support operating system and application under investigation.
Binary Translation mode	Uses default object code translation to dynamically convert target application and operating system code into new code that simulates the original code in a specific hardware configuration.

Table 8: Simulation modes in SimOS

SimOS also includes a detailed simulator (implemented with standard simulation techniques) that runs in a loop, fetching, decoding, and simulating the effects of instructions on the machine's register set, caches and main memory. The simulator includes a pipeline model that can record more detailed performance information at the cost of longer simulation time. Therefore, SimOS is able to provide hardware simulators ranging from very approximate to highly accurate models [6].

With all these features, SimOS can support highly realistic application workloads including those that require significant run time to get to a steady state operating point. The simulation models of SimOS are heavily instrumented to collect statistics about the simulated behaviour and map these statistics back to user defined groups such as processes, subroutines or transactions. With its highly detailed simulation models, SimOS can accurately model the components of today's complex computer systems [6].

Few projects that are using SimOS:

- IRAM (Intelligent RAM by UC Berkeley)
- SMI (Shared Memory Interface is a hardware distributed shared memory library (written in C) that uses SCI for messaging)
- CVM (Coherent Virtual Machine is a traditional distributed shared memory library that uses Ethernet for messaging)

2.7 Dimemas

Dimemas [7] is a trace driven performance prediction tool for message passing programs which enable users to develop and tune parallel applications by giving accurate prediction of their performance on the target machine architecture. Supported architectures include networks of workstations, Shared Memory Processors (SMP) and clusters. Figure 11 illustrates the Dimemas environment. Dimemas rebuilds the behaviour of a parallel program based on an input Dimemas trace file and information of the supported target machine architecture. Then, it generates a visualization trace file that is viewed using a visualization tool such as Paraver. The user can then analyse the performance and modify the message passing programs.

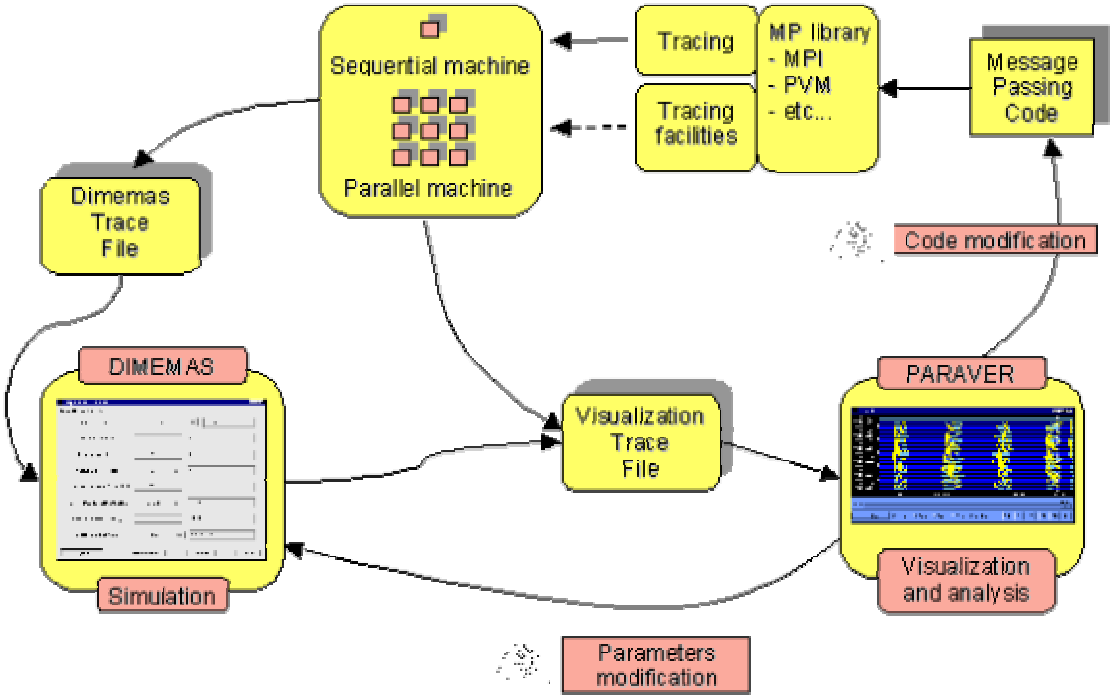

Figure 11: Dimemas Environment

Dimemas models the target machine architecture as a network of nodes where each node is an SMP connected to the network with a set of links and buses. Every node has a set of processors and local memory. L represents the number of links from a node to the network which restricts the number of messages moving in and out of the nodes. B represents the number of buses in the network which limit the number of messages using the network. The model of the target machine architecture defines information such as number of nodes, number of processors per node, network bandwidth, communication latency, number of input and output links and number of buses [7].

Projects that are using Dimemas:

- PMAC (Performance Modelling and Characterization)
- DiP (A parallel program development environment)

2.8 SvPablo

SvPablo [8] is a language independent performance analysis and visualization tool that supports data capture, analysis and presentation of sequential and parallel applications written in a number of programming languages (eg. C, Fotran, HPF). This enables the users to rapidly identify and resolve performance bottlenecks. Performance data is stored in a language independent and portable file format to support scalability such as addition of new metrics and support of new languages without having to alter the tool.

SvPablo computes performance analysis based on gathered application and hardware performance data. The SvPablo libraries support two types of performance instrumentation, namely automatic and interactive instrumentation to gather application data. Automatic instrumentation relies on the compiler or runtime system to insert points at which data should be captured automatically into the codes, whereas interactive instrumentation supports greater user control by allowing the user to specify the points [8].

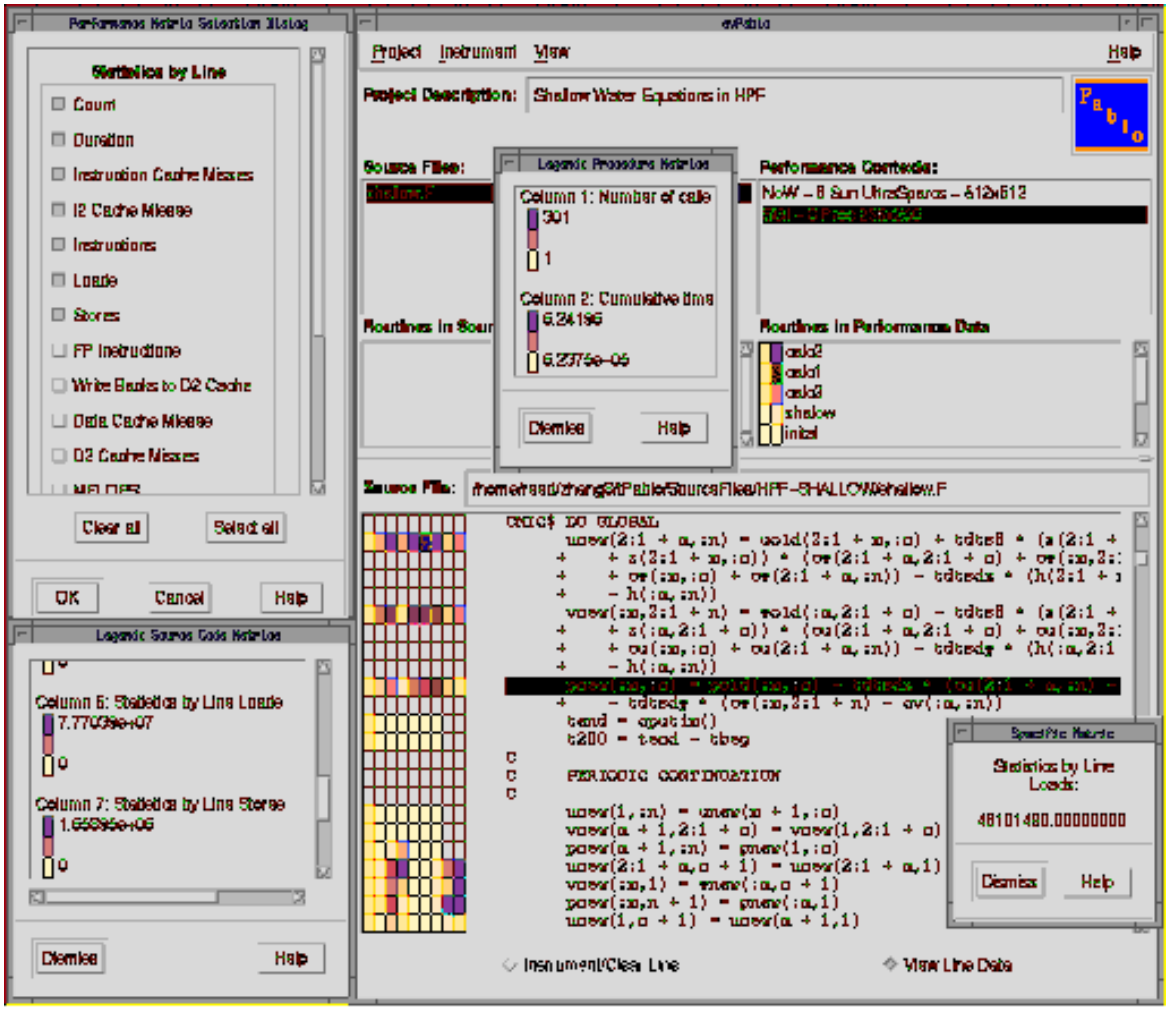

Figure 12: Graphical user interface of SvPablo

During the execution of the instrumented codes, hardware characteristics are also captured before computing and writing the statistics (rather than event traces) into summary files. Using statistics instead of detailed event traces

enables SvPablo to capture execution behaviour of codes that execute for hours and days on hundreds of processors. The files are then merged using a utility program before input into the graphical user interface. Figure 12 shows the SvPablo graphical user interface which display the code and performance data from an HPF program using colour-coded routine profiles [8].

The Pablo self-defining data format (SDDF) is used to define the data structures holding the analysis data and metrics. This SDDF enables the generality and extensibility required to add and modify a diverse and changing set of language independent performance metrics and measurement points, without affecting the graphical user interface [8].

2.9 MPISim

MPISim [9] is a library incorporated in the Message Passing Interface (MPI) standard to enable prediction of the performance of MPI programs based on architectural characteristics such as number of processors and message communication delays. MPISim assumes that the program has no I/O commands and simulates all collective communication functions in terms of point-to-point communication and all point-to-point communications are implemented using a set of four core non-blocking MPI functions: MPI_Issend (non-blocking synchronous send), MPI_Ibsend (non-blocking buffered send), MPI_Irecv (non-blocking receive) and MPI_Wait.

In order to enable the MPISim simulation to be run on a single processor machine, there is a need to modify an existing MPI program to support multithreaded execution. A pre-processor is provided with MPISim to automatically privatize permanent variables, changes each MPI call to MPISim call and implements various transformations needed to link the program with the MPISim library [9].

During simulation, each process in the MPI program is defined as a logical process (LP) in MPISim. Each LP has a message queue for each communicator of which LP is a member, a simulation clock and an ordered list (ordered by simulation timestamp) of the pending (send and receive) operations to be performed. Sequential code blocks are simulated through direct execution, while each call to a MPI communication function is translated to a corresponding MPISim function. The translation is done internally by MPISim to replace MPI functions to a set of four core non-blocking MPI functions as mentioned earlier [9].

MPISim produces a number of performance and behavioural metrics that can be used for analysis and comparison [9] as listed in Table 9.

Metric	Description
Application	Produces data related to the predicted performance of the target program, such as execution time, number and size of messages sent or number and type of I/O operations performed.
Simulation	Produces data specifying performance characteristics of the simulator code, such as actual simulator execution time, number and type of simulation protocol messages sent, or number of messages sent to simulator processes on other processors
System	Produces data detailing the simulated communication and file system configuration and behaviour, such as communication latency factors, number of compute nodes and I/O nodes, or number and type of disk operations.

Table 9: Metrics produced by MPISim

2.10 Ptolemy 2

Ptolemy 2 [10] consists of java-based libraries to design and simulate heterogeneous concurrent systems. It is built upon a component-based design methodology that supports hierarchical integration of a wide and extensible range of computation models to capture different specific domains. This hierarchical representation of heterogenous models preserves the understandability, manages complexity and encourages component reuse. The java packages of Ptolemy 2 are summarized in Table 10.

Package	Purpose
Kernel	Supports clustered hierarchical graphs which are collections of entities and relations between the entities.
Actor	Extends the kernel so that entities have functionality and can communicate via the relations.
Domain	Extends the actor by imposing models of computation on the interaction between the entities.
Other Packages	Supports many other specific packages including visual display of data, expression parser, data encapsulation, graph, math, matrix, vector and signal processing functions.

Table 10: Packages of Ptolemy 2

In Ptolemy 2, the components (known as actors) are defined by extending the actor package to execute tasks and communicate with other actors through ports. An actor can be atomic or composite. A composite actor is different from an atomic actor as it contains other actors (composite or atomic). A port can be an input, output or both [10].

A model of computation associated with a composite actor is implemented as a domain. A domain defines the communication semantics and execution order among the actors. Table 11 summarizes the wide variety of domains that have been implemented in Ptolemy 2. The communication mechanism is implemented using receivers. Examples of receivers include FIFO queues, mailboxes, proxies for global queue & rendezvous points. This clear separation of computation & communication enables reuse of actors [10].

Domain	Description
Continuous time (CT)	Models ordinary differential equations (ODE) extended to enable the handling of discrete events.
Discrete event (DE)	Enables actors to communicate through events placed on a continuous time line.
Synchronous Dataflow (SDF)	Allows components to consume and produce the same amount of input and output data sent through FIFO queues respectively.
Finite state machines (FSM)	Specifies precise sequencing of component states and transitions between the states.
Real time operating system (RTOS)	Allows designers to explore priority-based scheduling policies and their effects on real time software.
3D Visualization (GR)	Provides a component based infrastructure to display three dimensional graphics.

Table 11: Packages of Ptolemy 2

Figure 13 illustrates a pendulum control system modelled using Ptolemy 2. The system comprises of five actors: the pendulum, the controller, the network, an actor that represents other network components and a visualization actor [10].

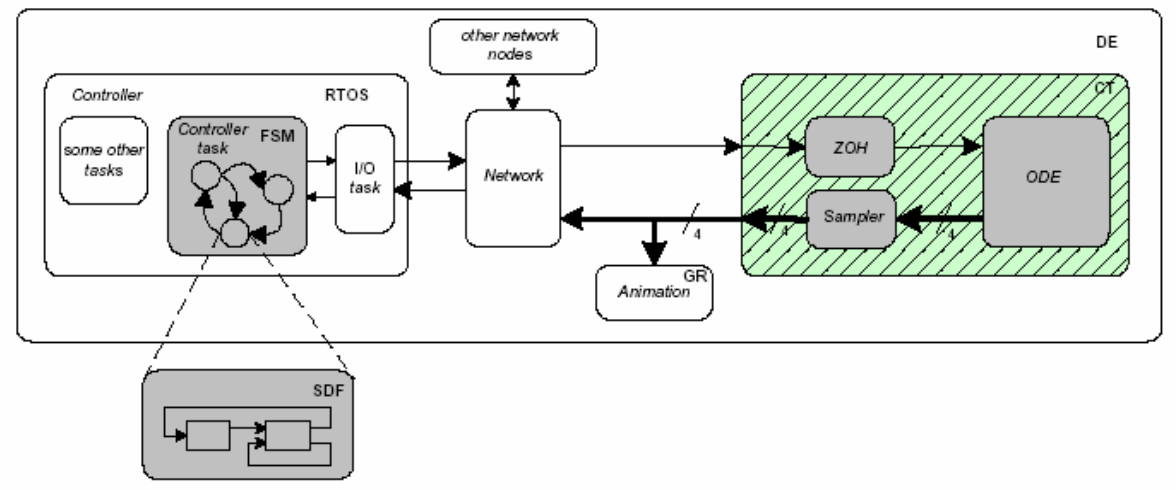

Figure 13: A pendulum control system

3. Comparison

In this section, the tools are compared to outline the core differences between them. The comparison is done based on different criteria to provide a better clarity and understanding of the tools.

3.1 Tool Classification

Simulation tools can be classified into two main categories depending on how aid in the construction of users' application simulations. The first category is library-based tools where the tool provides libraries to create the simulations. The second category is visual-based tools where the tool gives a graphical interface to reflect the simulation. The third category is the language programming used by these tools. Few of the tools could not be found, like MicroGrid and Dimemas. Some tools may belong to both categories. Table 12 shows the classification of the tools.

No	Tool	Library-based	Visual-based	Language used
1.	SimJava	Yes	Yes	Java
2.	Bricks	Yes	No	Java
3.	MicroGrid	Yes	No	C
4.	Simgrid	Yes	No	C
5.	GridSim	Yes	Yes (using SimJava)	Java
6.	SimOS	Yes	No	C
7.	Dimemas	Yes	Yes (using Paraver)	C
8.	SvPablo	Yes	Yes	C, Fortran 97, Fortran 90, HPF
9.	MPISim	Yes	No	C and MPI
10.	Ptolemy 2	Yes	Yes	Java

Table 12: Classifications of tools

3.2 Simulated Domain

No	Tool(s)	Simulated Domain
1.	SimJava	Process based discrete event simulation.
2.	Bricks MicroGrid Simgrid GridSim	The analysis and design of resource scheduling algorithms for grid resources in a grid computing environment.
3.	SimOS	Simulates complete computer systems, including computer architecture and operating system.
4.	Dimemas SvPablo MPISim	Analyse the performance of parallel programs.
5.	Ptolemy 2	Simulates behaviour of concurrent heterogenous components.

Table 13: Descriptions of tools in a simulated domain

3.3 Target Application Areas

No	Tool(s)	Target Application Areas
1.	SimJava	Used to model hardware and distributed software systems such as communication protocols and computer architectures.
2.	Bricks	Used to analyse and simulate various scheduling algorithms in global computing systems.
3.	MicroGrid	It supports the controlled emulation of Globus environment and this helps in studying resource management issues through the execution of real applications.
4.	Simgrid	Concerned with evaluation of algorithms for scheduling distributed applications in different grid settings.
5.	GridSim	Concerned with evaluation of algorithms for scheduling distributed applications in different grid scenarios. It is geared towards the Grid Economy— it supports the simulation of multiple competitive users and a wide variety of computing resources.
6.	SimOS	Concerned with simulating a complete computer system and studies all various aspects including hardware architecture, operating system and application programs.
7.	Dimemas MPISim	Used to evaluate the performance of message passing programs written in MPI.
8.	SvPablo	Used to evaluate the performance of sequential programs written in C and Fortran, and parallel programs written in HPF.
9.	Ptolemy 2	Simulates systems that comprises of heterogeneous components and sub-components.

Table 14: Descriptions of tools for suitable application areas

3.4 Design

SimJava provides a set of foundation classes for simulating discrete events. It includes facilities for representing simulation objects as animated icons and displaying dynamic time diagrams and charts [1].

Bricks provides simulation for resource allocation strategies and policies for multiple clients and servers as in global computing systems. On the other hand, Simgrid target scheduling algorithms for a single structured application, as opposed to scheduling policies for multi-user systems [4].

The major difference between MicroGrid and GridSim/Simgrid is that MicroGrid executes actual application code on the virtual Grid, and thus requires more time to complete the application, as compared to Simgrid running in a simulated grid. Another difference is that Microgrid simulations targets existing Globus applications whereas GridSim/Simgrid targets scheduling algorithm which work with an application model [4].

Simgrid is restricted to a single scheduling entity and time-shared systems and thus it is difficult to simulate multiple different users, applications, and schedulers operating in grid computing environment. GridSim addresses this by supporting space-based large-scale resources along with time-based resources in the grid environment [5].

SimOS provides modelling of complete computer systems through fast simulation of hardware and good control of level of abstraction [6].

Dimemas is a trace-driven simulator that rebuilds the simulation based on an input trace file obtained from traced execution of the message passing program on a platform different from the one to be evaluated. On the other hand, MPISim uses direct execution to obtain computational statistics of programs, so the host and target platforms have to be the same for accurate simulation results [7].

SvPablo is designed to be language independent and portable as it supports evaluation of applications written in different programming languages on sequential and parallel systems. It differs from Dimemas as it performs run-time summarization instead of reading from a trace file and the data capture and visual presentation components are integrated into the same graphical performance browser. In addition, SvPablo uses statistics not detailed event trace. This enables SvPablo to capture execution behaviour of codes running for hours on many processors [8].

Ptolemy 2 is built upon a component-based design methodology that hierarchically integrates multiple models of computation to capture different design perspectives. It also incorporates a 3-D animation package for visualization of control results [10].

4. Conclusion and Further Work

This paper provides a topical overview of the recent progress of parallel and distributed simulation tools by outlining some of the many tools currently in research and conducts a brief survey on these tools. It reinforces the belief that simulation does help in the modelling and development of real-world processes, especially in the area of parallel and distributed systems for high performance computing. Simulation tools are used to model and predict the behaviours of the entities and the interactions among them in a parallel and distributed computing environment.

Future work would include a more detailed analysis and comparisons of the various tools. In addition, we would like to conduct some experiments using these tools for particular relevant criteria.

5. Acknowledgement

The technical information and diagrams of simulation tools described in this paper is derived from articles published by their authors. We would like to thank them for their outstanding work. We thank Srikumar Venugopal for his comments on the paper.

6. References

[1] F. Howell and R. McNab. *SimJava: A Discrete Event Simulation Package For Java With Applications In Computer Systems Modelling*. First International Conference on Web-based Modelling and Simulation, San Diego, CA, Society for Computer Simulation, Jan 1998.

[2] A. Takefusa, S. Matsuoka, H. Nakada. *Overview of a Performance Evaluation System for Global Computing Scheduling Algorithms*. 8th IEEE International Symposium on High Performance Distributing Computing (HPDC8), 1999.

[3] H. J. Song, X. Liu, D. Jakobsen, R. Bhagwan, X. Zhang, K. Taura, A. Chien. *The MicroGrid: a Scientific Tool for Modelling Computational Grids*. IEEE Supercomputing (SC2000), Nov. 4-10, 2000, Dallas, USA.

[4] H. Casanova. *Simgrid: a Toolkit for the Simulation of Application Scheduling.* Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid2001), May 15-18, 2001, Brisbane, Australia.

[5] R. Buyya and M. Murshed. *GridSim: A Toolkit for the Modelling and Simulation of Distributed Resource Management and Scheduling for Grid Computing*. The Journal of Concurrency and Computation: Practice and Experience (CCPE), Wiley Press, May 2002.

[6] M. Rosanblum, S. A. Herrod, E. Witchel, A. Gupta. *Complete Computer System Simulation: The SimOS Approach*. IEEE Parallel and Distributed Technology, 1995.

[7] S. Girona, J. Labarta, R. M. Badia. *Validation of Dimemas Communication Model for MPI Collective Operations*. 7th EuroPVM/MPI, 2000.

[8] L. DeRose and D. A. Reed. *SvPablo: A Multi-Language Architecture-Independent Performance Analysis System*, International Conference on Parallel Processing (ICPP99), 1999.

[9] S. Prakash and R. L. Bagrodia. *MPI-SIM: Using Parallel Simulation to Evaluate MPI Programs*, 1998 Winter Simulation Conference (WSC98), 1998.

[10] J. Eker, C. Fong, J. W. Janneck, J. Liu. *Design and Simulation of Heterogeneous Control Systems Using Ptolemy II*, IFAC Conference on New Technologies for Computer Control (NTCC01), 2001.

[11] J. Liu, D. Nicol, B. Premore, A. Poplawski. *Performance Prediction of a Parallel Simulator*, Parallel and Distributed Simulation Conference (PADS99), 1999.

[12] A. J. C. van Gemund. *Performance Prediction of Parallel Processing Systems: The Pamela Methodology*, ACM International Conference on Supercomputing, 1993.

[13] S. Gilmore and J. Hillston. *The PEPA Workbench: A tool to Support a Process Algebra-based Approach to Performance Modelling*, 7th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation, 1994.

[14] D. F. Kvasnicka, H. Hlavacs, C. W. Ueberhuber. *Simulating Parallel Program Performance with CLUE*. International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), Society for Modelling and Simulation, San Diego, 2001.

[15] N. Senin, R. Groppetti, A. Rossi and D. Wallace. *Integrating Manufacturing Simulation Tools Using Distributed Object Technology*. 4th IEEE/IFIP International Conference on Information Technology for Balanced Automation Systems in Production and Transportation, Berlin, September 2000.

[16] R. Suri and M. Tomsicek. *Rapid Modeling Tools for Manufacturing Simulation and Analysis.* Proceedings of the 20th Conference on Winter Simulation, San Diego, 1988.

[17] H. Goncalves, M. Oliviera and A. Patricio. *The Use of a Simulation Tool In Order to Obtain the Thermal Performance of Passive Solar Houses In Portugal*. Proceedings of Building Simulation, Vol. 2, 1997.

[18] J. Nel, J. Roux and P. Schneider. *CODYBA: A Design Tool for Buildings Performance Simulation.* Proceedings of Building Simulation, Vol. 1, 2001.