Automatic Modelling and Grid Scheduling MPI Applications

WCGA - V Workshop on Grid Computing and Applications

Daniele S. Jacinto Ricardo A. Rios Hélio C. Guardia

Departament of Computer Science
Federal University of São Carlos
{daniele_jacinto, ricardo_rios, helio}@dc.ufscar.br

29th May 2007

Overview

Why

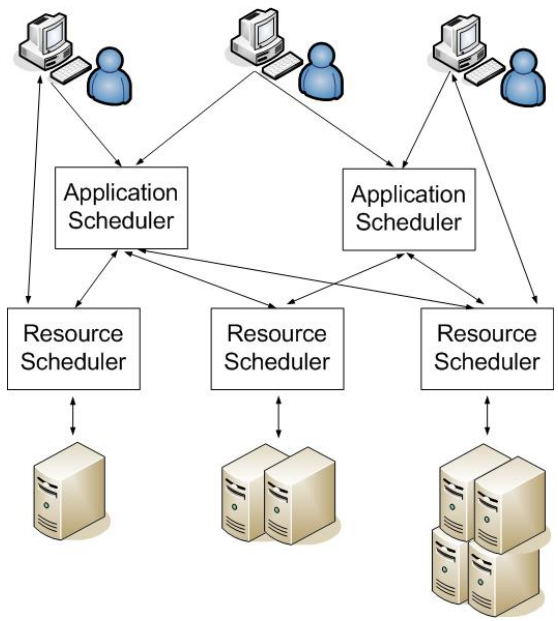
- Most scheduling algorithms for computacional grids rely on an **application model** represented by a **directed acyclic graph** (DAG).
- The **obtainment of DAGs** for real applications is not simple.
- Many MPI programs present a **cyclic communication model** which prevents **scheduling algorithms** to be employed as they **recursively traverse** the graphs.

How

- A **monitored execution in a cluster** is used to detect dependencies. Comunication **cycles are then eliminated** to produce an appropriate DAG. **Scheduling is performed** in a Globus grid.

Introduction

- The multitude of resource motivates **migrating an application to a grid**.
- Creating a **single scheduler** to manage a grid, over different administration domains, **is a complex task**.
- The scheduling in Grid environment is usually divided into an **Application** and **Resource components**, forming a multi-layer **Resource Management System** (RMS).



Grid Scheduling

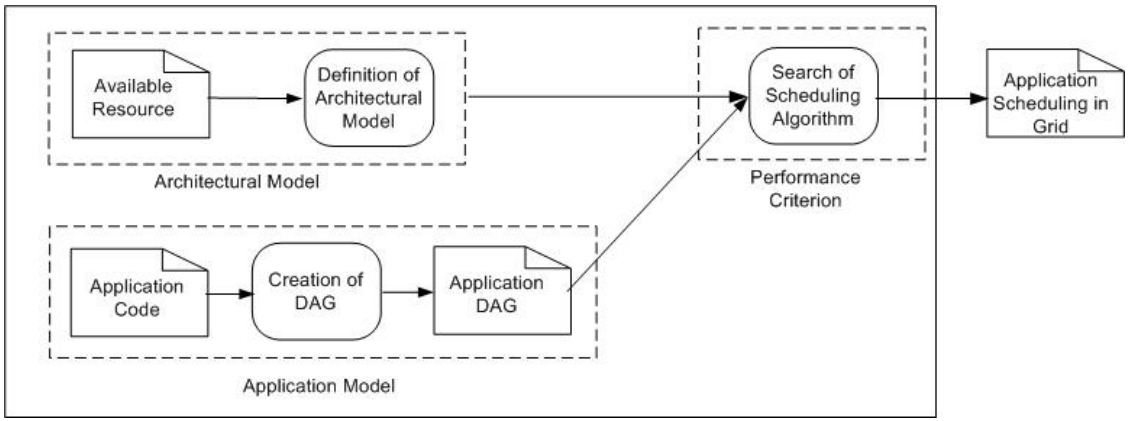
Grid Scheduling

Application Scheduler

Based on an **application model**, on an **architectural model** and on a **performance criterion**

Resource Scheduler

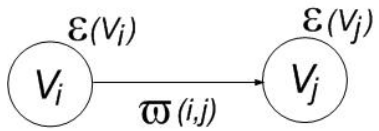
Based solely on **hardware prerequisites** (necessary memory, type of CPU, etc)



Application Scheduler Structure

Application Scheduler

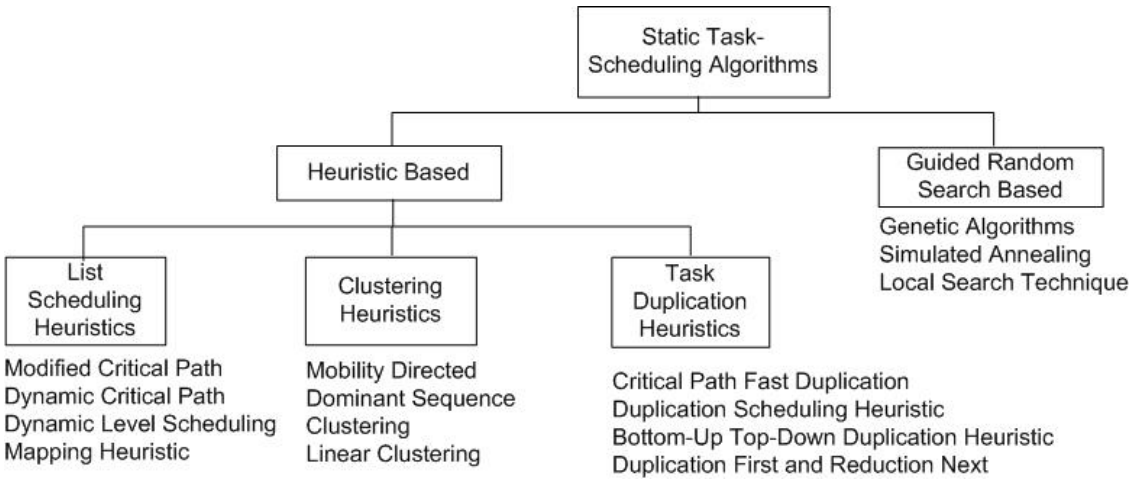
- **Application Model**
  The application model represents the **tasks of an application** (V_i, V_j), the **communications among them**, the **computational costs** of those tasks $(\varepsilon_{(V_i)}, \varepsilon_{(V_j)})$ and of the application, and the **communication weights** $\omega_{(i,j)}$.



Application Model

Application Scheduler

- **Architectural Model**
  Number of processors, channel communication and their latency, and heterogeny of the processors
- **Performance Criterion**



Classification of static task-scheduling algorithms

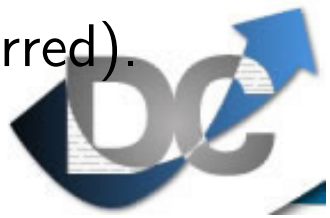
Scheduling communicating parallel applications

Our approach

1. **Application monitoring**: determine tasks and communications
2. **Graph generation**: process logs to produce graph information file
3. **Communication aggregation**: summarize inter node communications
4. **DAG generation**: elimination of cycles using virtual nodes
5. **Scheduling**

DAG Generation of MPI Applications

- Creating a **realistic model of the application** behavior can be complex and is usually left to the **developer herself**.
- **Knowledge of an applications** features can generally be obtained by **monitoring**.
- The monitoring during the **execution of the application in clusters**, determines the **computational cost of a task** (total processing time) and the **weight of each communication** (total communication time and amount of data transferred).



Monitoring

- **This work analyzes the log** files generated by the **Intel Trace Collector** (ITC) tool.
- Each line of the tracefile contains a single event or command written according to the syntax:
  [time] ⟨**command**⟩ **[parameter [parameters]]**.

Classification of ITC commands

(i) **Configuration commands**: describe important tracing parameters used for visualization,

(ii) **Event commands**: events that occurred during the execution of the program.

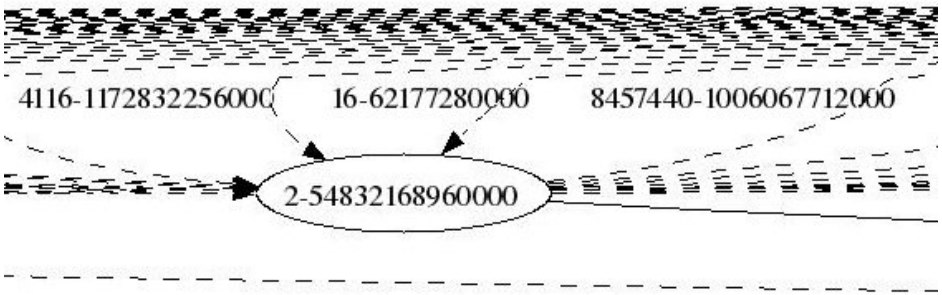
Monitoring

Parts of Tracefile from IS NAS Class C

CLKPERIOD 1.2207E-13
EVENTBITS 13
DURATION 0 54913662976000
COMDEF 2 4 0:3:1 NAME "COMM_WORLD"
...
NCPUS 4
...
DEFSTATE 202 ACT 10 "MPI_Send"
DEFSTATE 258 ACT 10 "MPI_Reduce"
DEFSTATE 224 ACT 10 "MPI_Wait"
DEFSTATE 506 ACT 5 "User_Code"
...
466944000 EXCHEXT CPU 4 DOWNTO 506
18235392000 EXCHEXT CPU 3 DOWNTO 506
46358528000 EXCHEXT CPU 1 DOWNTO 506
...
54722674688000 EXCHEXT CPU 1 DOWNTO 202
54722674688000 SENDMSG 2 1000 FROM 1 TO 2 LEN 4 FUNCTION 202
54724583424000 EXCHEXT CPU 2 UPTO 506
54724583424000 RECVMSG 2 1000 BY 2 FROM 1 LEN 4 FUNCTION 224
54769082368000 EXCHEXT CPU 3 DOWNTO 213
...
54913523712000 EXCHEXT CPU 1 UPTO 506
54913523712000 EXCHEXT CPU 1 UPTO NOACT

Processing of the tracefile

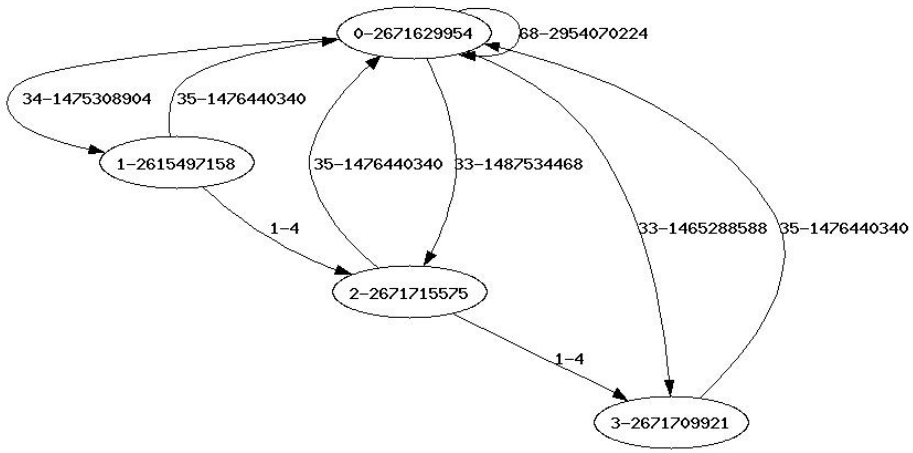
- **Obtaining the task execution time**
  46358528000 EXCHEXT CPU 1 DOWNTO 506
  ...
  54913523712000 EXCHEXT CPU 1 UPTO NOACT
- **Obtaining communications and their weights**
  53411512320000 SENDMSG 2 1000 FROM 2 TO 3 LEN 4 FUNCTION 202
  ....
  54770655232000 RECVMSG 2 1000 BY 3 FROM 2 LEN 4 FUNCTION 224
- **Graphical view of node 2 before summarizing the communications**



Graph Generation

Graphviz tool input file - generated through the tracefile analysis

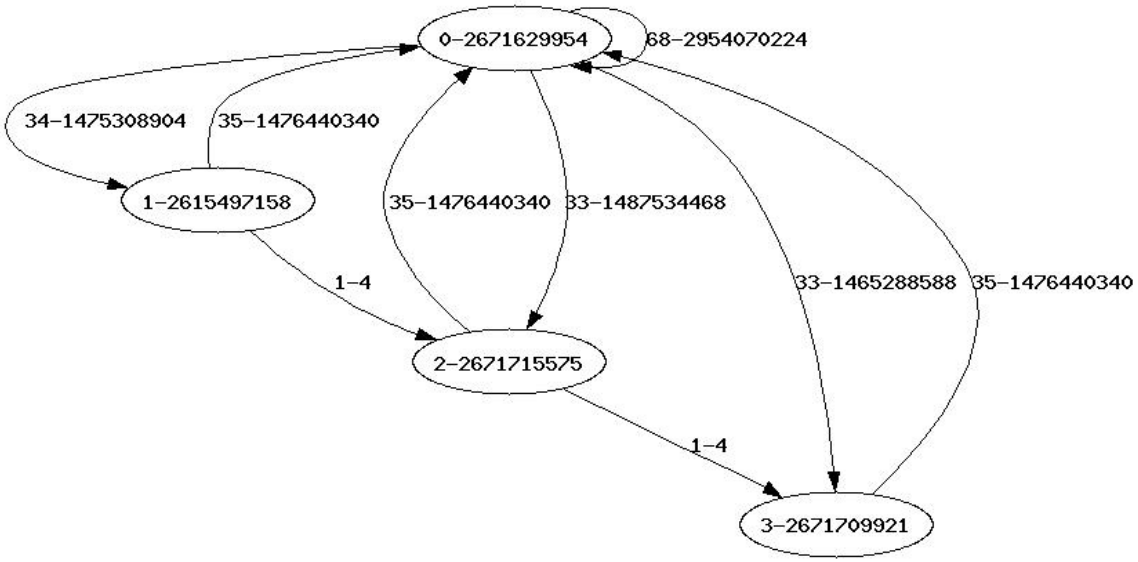
```
digraph G {
"n0" [ label = "0-2671629954"];
"n1" [ label = "1-2615497158"];
"n2" [ label = "2-2671715575"];
"n3" [ label = "3-2671709921"];
"n0" → "n0" [ label="68-2954070224-5053621283" ];
"n0" → "n1" [ label="34-1475308904-1971805275" ];
"n0" → "n2" [ label="33-1487534468-1997653470" ];
"n0" → "n3" [ label="33-1465288588-1857006648" ];
"n1" → "n0" [ label="35-1476440340-1971815362" ];
"n1" → "n2" [ label="1-4-73459617" ];
"n2" → "n0" [ label="35-1476440340-2008094538" ];
"n2" → "n3" [ label="1-4-10449665" ];
"n3" → "n0" [ label="35-1476440340-1857022819" ];}
```



- The original graph file had **561 lines** that needed **17 seconds** to be scheduled.
- The condensed graph file had **31 lines** that needed **0.00001 seconds** to be scheduled.

Graph Generation

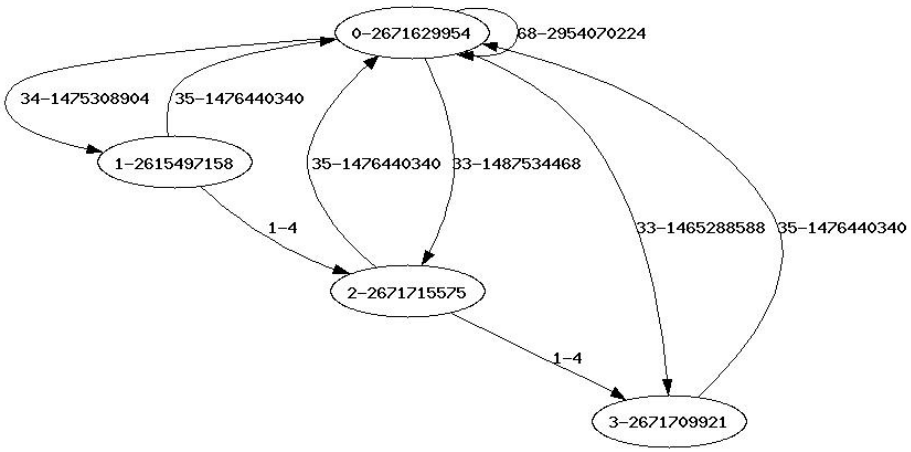
- The **total execution time of node 1 is 2615497158** μ**s**
- Node **0 sends 34 messages to node 1** totaling **1475308904 bytes**
- The elapsed time of the communications between these nodes is 1971805275 μs. ("n0" $\rightarrow$ "n1" [label="34-1475308904-**1971805275**"])
- The **graph is not yet adequate** for scheduling, however, as it **contains cycles**



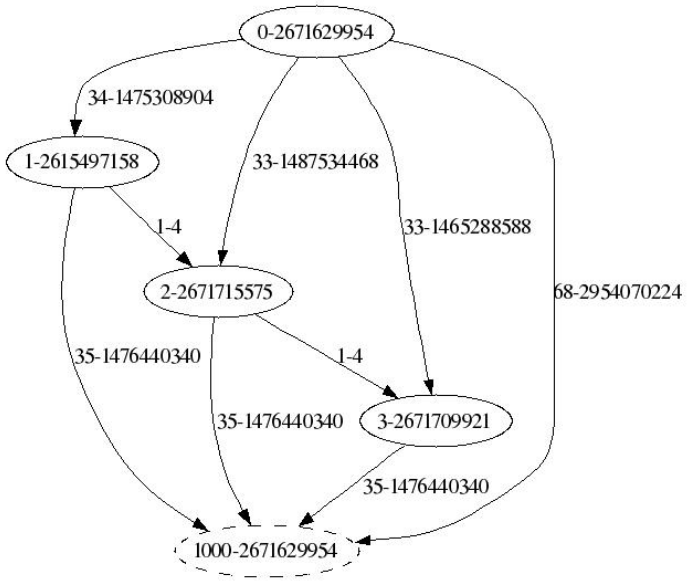
Cycle Removal

- MPI applications usually present some **sort of coordination among the nodes** and it **prevents most scheduling algorithms** to be employed as they **recursively traverse the graphs** to prioritize the tasks.
- We use a **depth-first analysis** in the produced graph **to search for communication cycles**. When all the edges of a node have been explored, the algorithm **backtracks to analyze the other edges** of the current parent node. **Vertexes are marked during the search** to indicate that they were visited.
- If a cycle occurs, a new vertex is created, called a **virtual node**, and all vertexes that previously joined v_1 to v_2 will now be directed to the virtual vertex, thus **eliminating the cycle**.

Cycle Removal



Condensed graph of NPB IS Class C



NPB Class C after the removal of cycles

Results

1. Evaluation of the Graph Produced
2. Analysis of the Cycle Removal Phase
3. Evaluation of the MPI Application Schedule

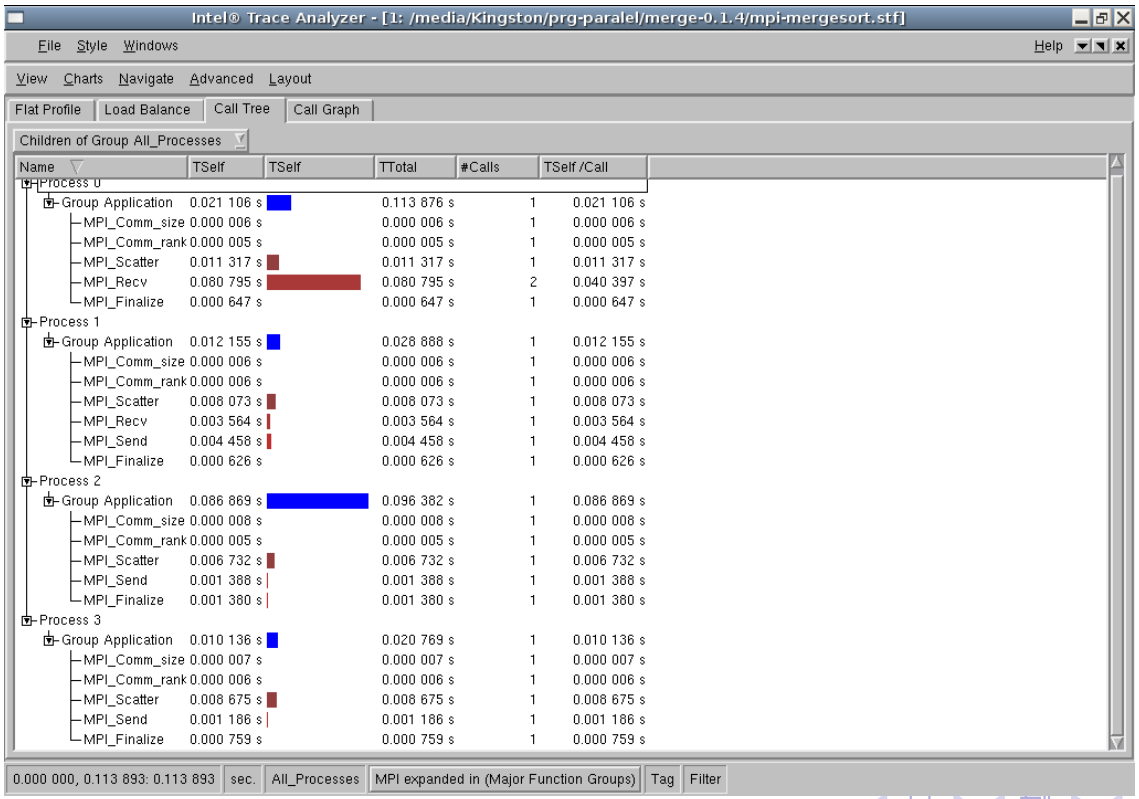
Results

Evaluation Process

1. Compilation of a target application linked with Trace Collector library.
2. Execution in a cluster.
3. The log generated by Trace Collector is analyzed and the relevant informations are extracted.
4. The corresponding task graph is modeled.
5. The task graph is traversed for removing existing cycles.
6. DAG for the application is produced.
7. The scheduling process is applied.

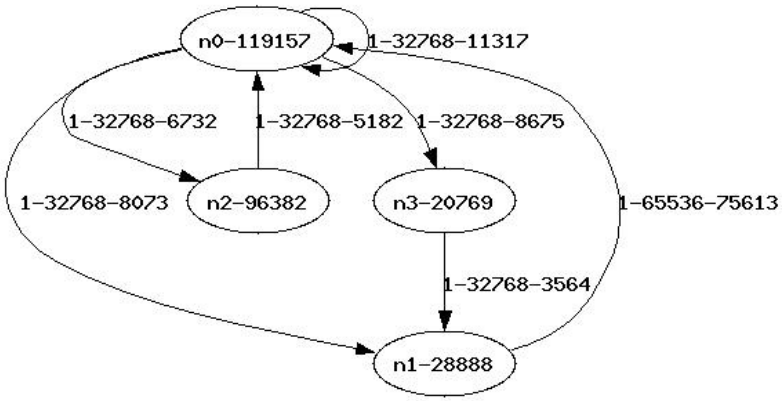
Evaluation of the Graph Produced

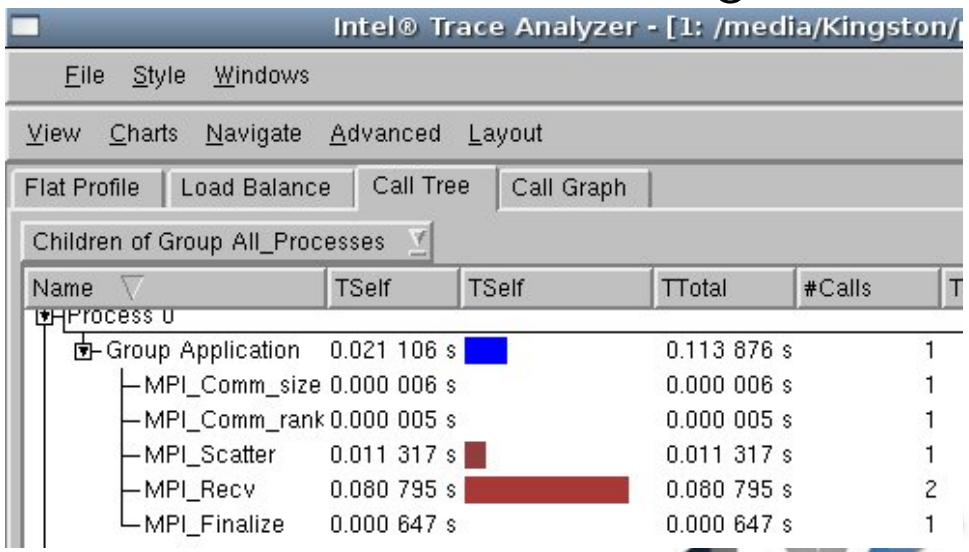
The graph generated for **MergeSort** application was analyzed using **Intel Trace Analyzer** and the results were compared with the DAG generated by our mechanisms.



Evaluation of the Graph Produced

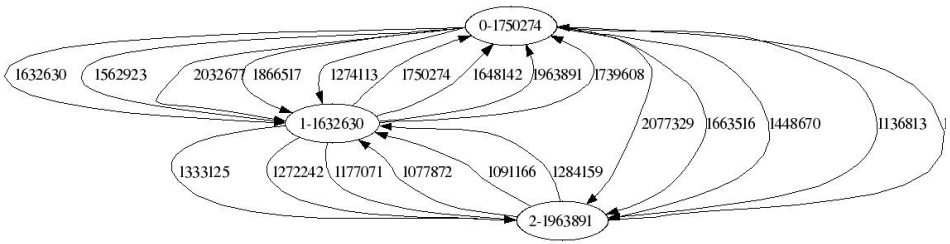
Node **0 receives two messages**, one from **node 2** and other from **node 1**. In the **Trace Analyzer** data, node 0 has **two Recv communications totaling 80795** μ**s** that correspond to the summation of both messages.



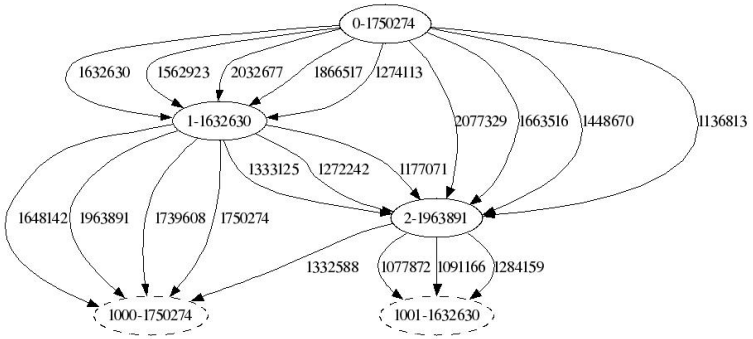


Analysis of the Cycle Removal Phase

- **Analysis of the cycle removal mechanism** using graphs collected from **real MPI applications** and also with **randomly generated graphs**.
- The **random graphs** were obtained using the **GTGraph** that is a synthetic graph generator suite developed for the DIMACS Shortest Paths Challenge.



original graph



after cycle removal (8)

Evaluation of the MPI Application Schedule

- Cluster environment was composed by **three resources** with a total of **11,940.04 bogomips and 1.5 GB RAM** connected through a dedicated Fast Ethernet network
- The grid environment was composed by **six resources** with **24,924.55 bogomips, 3.5 GB RAM**. The grid resources were connected through a Fast Ethernet network.

NAS Benchmark			
Execution Time (sec)			
	Class A	Class B	Class C
Cluster	14.86	60.89	15889.22
Grid	10.33	38.90	2162.26

Execution Time of NAS Benchmark

Evaluation of the MPI Application Schedule

Another result: Same network, more resources.

- **Application:** a matrix multiplication with dimension of 4096 elements and NP 5.
- **Cluster:** composed by 32 dual-core nodes with 4788 bogomips (**153,216 bogomips**) and 1 GB RAM on light load condition. The network communication uses FastEthernet tecnology
- **Grid:** a simulated grid environment (GridSim), with 24 resources, summing **666,047 bogomips** interconnected through a FastEthernet network. This grid has 3 routers and spawns over 3 different administrative domains.

The execution time on a **cluster was 1,555.621 seconds** and on a **grid was approximately 1,118.433 seconds** with a **random search of resources**. Using the **application scheduler** the execution time on a grid was **271.323 seconds**.

Conclusions

1. The use of the **DAGs allowed more effective scheduling in the grid**, and the results show the approach was successful due to the appropriate host selection.
2. The mechanism we have developed **automatically generates DAGs of MPI applications simplifying their scheduling in a grid** while still producing enhanced execution.
3. As the automatic DAG creation of an application is enabled by this work, we believe **cluster applications can more easily be adapted for execution in grid environments**.

Future Works

Adequate the scheduling of (scalable) parallel applications to run on a larger number of grid nodes.

Automatic Modelling and Grid Scheduling MPI Applications

WCGA - V Workshop on Grid Computing and Applications

Daniele S. Jacinto, Ricardo A. Rios, Hélio C. Guardia
Departament of Computer Science
Federal University of São Carlos
{daniele_jacinto, ricardo_rios, helio}@dc.ufscar.br