Scheduling on the Grid
Grid Scheduling Architecture

User Application → Grid Scheduler

Grid Scheduler → Grid Information Service

Local Resource Manager

Single CPU (Time Shared Allocation)

SMP (Time Shared Allocation)

Clusters (Space Shared Allocation)
Task Scheduling Problem

- 1000 tasks to complete
  - E.g. parameter study
- Deadline
- Cost
Parameter Study Tool

Compose, submit, and play

<table>
<thead>
<tr>
<th>East</th>
<th>West</th>
<th>North</th>
<th>South</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>30</td>
<td>40</td>
</tr>
<tr>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
</tr>
<tr>
<td>30</td>
<td>40</td>
<td>50</td>
<td>60</td>
</tr>
<tr>
<td>40</td>
<td>50</td>
<td>60</td>
<td>70</td>
</tr>
<tr>
<td>50</td>
<td>60</td>
<td>70</td>
<td>80</td>
</tr>
</tbody>
</table>

1st Qtr 2nd Qtr 3rd Qtr 4th Qtr
Backward Pricing

- Business Administration Department, University of Vienna
- Backward induction algorithm to compute the price of an interest rate dependent financial product, such as a variable coupon bond
- Parameters varied
  - the coupon bond (0.01 : 0.1 : 0.001)
  - the number of time steps (5 : 60 : 5)
  - the length of one time step (1/12 : 1 : 1/12)
- 1481 experiments
Total Price Visualisation
Task Scheduling

How to map all 1481 tasks on 300 machines such that the overall completion time is minimised?

Search and optimisation problem
- 1481 experiments
- Each experiment can be run on 300 processors
- \(300^{1481}\) possible solutions!

Heuristics must be used to find “good” solutions
The Scheduling Problem

- **Hypothesis**
  - N tasks
  - M machines
  - Execution time of each task on each machine

- **Goal**
  - Find the optimum mapping of the N tasks on the M machines that produces the fastest completion time

- **Solution**
  - Finding the optimum requires to evaluate all search space points
  - There are $M^N$ possible mapping
  - This is impossible in real time

- **NP-complete optimization problem**
  - Cannot be solved in polynomial time
  - Requires heuristics to find approximate "good" solutions
Optimization Framework

- Scheduling as an instantiation

1. Parameterized Application (task, machine)
2. Heuristic Search Engine
3. Solution (schedule)
4. Objective Function
5. Prediction Function
6. Execution Model

"Best" Solution (schedule)
Sample Objective Functions

- execution time (makespan)
  \[ S = \min_{\forall CPU \in \text{Grid}} \{ T_{\text{seq}}(CPU) \} \]

- speedup
  \[ E = \frac{S}{\sum_{\forall CPU \in \text{Grid}} S_{CPU}}, \quad S_{CPU} = \min_{\forall CPU \in \text{Grid}} \frac{T_{\text{seq}}(CPU)}{T_{\text{seq}}(CPU)} \]

- efficiency + execution time = throughput
- communication time
- load balance
  \[ L = \frac{\text{Avg}(T_{CPU})}{\text{Max}(T_{CPU})} \]
- Scheduling chart
- How tasks are mapped to machines over time
- A timetable of machine usage

**Gantt Chart**

- **M1**: T2, T4, T6
- **M2**: T1, T3, T8
- **M3**: T5, T7

Timeline
Homogeneous versus Heterogeneous

- How long will a task execute on a processor?
  - Time = Work / Speed
  - Computation: Time = Flops / Clock_speed
  - Network: Time = Data_size / Bandwidth

- Homogeneous systems
  - Each task has a cost that represents the execution time

- Heterogeneous systems
  - ETC Matrix = Expected Time to Compute
  - 2 types used in simulation
    - Consistent: if a machine is faster for a task t1, it is also faster for a task t2
    - Inconsistent
Performance Prediction

- Regression functions based on historical measurements
- Application specific analytical models
  - \( T = \frac{\text{Work}}{\text{speed}} \)
  - \( W_{\text{LAPW1}} = 7AN^2 + N^3 \)
    - \( A \) = number of atoms
    - \( N \) = matrix size
    - \( 7 \) = scaling factor
  - Speed: machine ranking
- Naive theoretical approximations
  - \( T_{\text{CPU}} = \frac{\text{Flops}}{\text{Clock}} \)
  - \( T_{\text{NET}} = \frac{\text{Size}}{\text{Bandwidth}} \)
**O(m) Heuristics (I)**

- **Minimum Execution Time (MET)**
  - Schedule a task on the machine that executes it fastest
  - Do not consider queuing time
  - Fast machines will be overloaded and slow machines will be idle

- **Minimum Completion Time (MCT)**
  - The earliest completion time on a machine
  - I/O transfer time + MET + machine ready time

- **Switching Algorithm (SA)**
  - Use MET and MCT heuristics in a cyclic fashion
  - Use MET at the expense of load balance until a given threshold
  - Then use MCT to smooth the load across the machines
O(m) Heuristics (II)

- **K-Procent Best (KPB)**
  - Considers a subset of only km/100 best machines based on the execution time of a task, where:
    - m is the total number of machines
    - \( 100/m \leq k \leq 100 \)
  - Assigns a task to the machine from the subset that delivers ECT
  - Avoids putting the current task onto a machine which might be more suitable for some yet-to-arrive tasks
  - \( k = 100/m, \ KPB = \text{MET} \)
  - \( k = 100, \ KPB = \text{MCT} \)

- **Opportunistic Load Balancing (OLB)**
  - Assign the task to the first available machine
  - Good for heavily used environments
Sample Simulation Results

Figure 4. Makespan for the on-line heuristics for inconsistent HiHi heterogeneity.

Figure 6. Makespan of the on-line heuristics for semi-consistent HiHi heterogeneity.
Min-Min Heuristic

- Determine the MCT for all the tasks
- Schedule only the task with the minimum MCT and reconsider the rest
- Execute the smallest jobs first to get done as many of them as possible
  - Do the easy things first
- Good when the number of short and long jobs are evenly distributed
- Bad when there are many more shorter tasks than long tasks

<table>
<thead>
<tr>
<th></th>
<th>M1</th>
<th>M2</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>M2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
Min-Min Algorithm — $O(n^2m)$

```
algorithm min-min(task_list, Grid)
GanttChart = Ф
while task_list ≠ Ф
    foreach task ∈ task_list
        (machine, time) = MCT(task, Grid, GanttChart)
        MCT_list = MCT_list ∪ (task, machine, time)
    end foreach
    (task_s, machine_s) = min_{time} MCT_list
    GanttChart = GanttChart ∪ (task_s, machine_s)
    task_list = task_list - task_s
end while
return GanttChart
```
**Min-Max Heuristic**

- Determine the MCT for all the tasks
- Schedule only the task with the maximum MCT and reconsider the rest
- Execute the biggest jobs to get around the Min-Min trap
  - Do the hard things first
  - Good when there are few hard things and lots of easy things

\[
\begin{array}{c|cccccc}
\text{M1} & 2 & 2 & 2 & 2 & 2 & 2 \\
\text{M2} & & & & & & 12 \\
\end{array}
\]
Min-Max Algorithm — \( O(n^2m) \)

**algorithm** min-max(task_list, Grid)

\[ GanttChart = \emptyset \]

**while** task_list \( \neq \emptyset \)

**foreach** task \( \in \) task_list

\[ (\text{machine}, \text{time}) = \text{MCT}(\text{task}, \text{Grid}, GanttChart) \]

\[ \text{MCT\_list} = \text{MCT\_list} \cup (\text{task}, \text{machine}, \text{time}) \]

**end foreach**

\[ (\text{task}_s, \text{machine}_s) = \max_{\text{time}} \text{MCT\_list} \]

\[ GanttChart = GanttChart \cup (\text{task}_s, \text{machine}_s) \]

\[ \text{task\_list} = \text{task\_list} - \text{task}_s \]

**end while**

**return** GanttChart
Sufferage Heuristic

- Determine the MCT for all the tasks
- Schedule first jobs that will suffer most if not executed on a certain machine
- Suffer = fastest - second fastest

\[ \text{Min-Min} \]

\[ \text{Sufferage} \]

<table>
<thead>
<tr>
<th></th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>40</td>
<td>48</td>
<td>134</td>
<td>50</td>
</tr>
<tr>
<td>T2</td>
<td>50</td>
<td>82</td>
<td>88</td>
<td>89</td>
</tr>
<tr>
<td>T3</td>
<td>55</td>
<td>68</td>
<td>94</td>
<td>93</td>
</tr>
<tr>
<td>T4</td>
<td>52</td>
<td>60</td>
<td>78</td>
<td>108</td>
</tr>
</tbody>
</table>
Sufferage Algorithm

```plaintext
algorithm sufferage(task_list, Grid)
GanttChart = Ф
while task_list ≠ Ф
    foreach task ∈ task_list
        (machine, time) = MCT(task, Grid, GanttChart)
        (machine2, time2) = 2nd_MCT(task, Grid, GanttChart)
        MCT_list = MCT_list ∪ (task, machine, time2 - time)
    end foreach
    { (task_s, machine_s) } = { max \text{time} | ∀ machine_s ∈ Grid MCT_list }
    GanttChart = GanttChart ∪ { (task_s, machine_s) }
    task_list = task_list - { task_s }
end while
return GanttChart
```
Task Stream Scheduling

Schedule(stream of jobs) {
  1. compute the next scheduling event
  2. create a Gantt Chart $G$
  3. for each task currently executing
  4. estimate its completion time
  5. fill in the slots in $G$
  6. select a subset of the tasks that have not started execution
  7. schedule them using an appropriate heuristic
  8. convert $G$ into a plan
}

- Assign ages to jobs to avoid starvation
  - Increase priorities of old jobs
Workflow Application

- **Workflows** = loosely-coupled set of activities organized in a graph
  - Coordination of off-the-shelf software components
  - Control flow dependencies
  - Data flow dependencies
  - Fits the model of loosely-coupled set of Grid resources

- **Models**
  - Directed Acyclic Graphs
  - Directed Graphs
    - Requires variables to implement loops
I. Institute for Material Chemistry, TU Vienna
   - [http://www.wien2k.at](http://www.wien2k.at)

II. Electronic structure calculations of solids using density functional theory

III. Application specific analytical models
   - \( T = \text{Work} / \text{speed} \)
   - \( W_{\text{LAPW1}} = 7 A N^2 + N^3 \)
     - \( A = \text{number of atoms} \)
     - \( N = \text{matrix size} \)
     - \( 7 = \text{scaling factor} \)
   - \( W_{\text{LAPW2}} = 10\% \ W_{\text{LAPW1}} \)
   - \( W_{\text{case.vector}} = 200A N \)
Invmod

- Institute of Hydraulic Engineering
  - [http://dps.uibk.ac.at/~marek/projects/invmod_wasim](http://dps.uibk.ac.at/~marek/projects/invmod_wasim)

- The Water Flow and Balance Simulation Model
  - WaSiM-ETH (Zurich)
Galaxy Clusters

- Institute of Astrophysics
- [http://astro.uibk.ac.at/](http://astro.uibk.ac.at/)
Workflow Scheduling

- Mapping of workflow activities onto the computational Grid resources
- NP-complete optimization problem
- Requires heuristics to find good solutions
- The independent task heuristics can be applied too
  - Have to consider the control flow dependencies
Objective Function

Gantt Chart

Timeline

Machine

M1

JS3  JS2  JS7

M2

JS1  FT4

JS8

M3

JS5  FT6

M2

JS8

M1

JS2  M1

JS3  M1

FT4  M2, M3

JS5  M3

FT6  M3, M2

JS7  M1

JS8  M2
List Scheduling Algorithms

- **DAGs in homogeneous environments**
- **Sorts the nodes of a DAG according to their cost**
- **T-level (ascending order)**
  - Length of the longest path from the entry node to node \( n \) (top level)
  - Scheduling two dependent task on the same CPU zeroes the communication dependency cost
- **B-level (descending order)**
  - Length of the longest path from the node \( n \) to the exit node (bottom level)
T-Rank and B-Rank

<table>
<thead>
<tr>
<th></th>
<th>T-rank</th>
<th>B-rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>N1</td>
<td>0</td>
<td>23</td>
</tr>
<tr>
<td>N2</td>
<td>6</td>
<td>15</td>
</tr>
<tr>
<td>N3</td>
<td>3</td>
<td>14</td>
</tr>
<tr>
<td>N4</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>N5</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>N6</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>N7</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>N8</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>N9</td>
<td>22</td>
<td>1</td>
</tr>
</tbody>
</table>
List Scheduling Algorithm

- Homogeneous environments
- Execution Time
  - A=7, B=11, C=4, D=9
- Data Transfer
  - AB=5, AC=3, BD=6, CD=2
- 2 phases
  - Ranking phase: assign rank values to tasks
  - Mapping phase: schedule the tasks in the order given by the ranks
- Scheduling list (B-level): A, B, C, D
HEFT Algorithm

- **Heterogeneous Earliest Finish Time**
  - Extension of the list scheduling algorithm for heterogeneous environments

- **Weighting phase**
  - Node / edge weight = mean values across all machines / data links

- **Ranking phase**
  - Rank tasks according to the weights
  - B-rank (see previous slide)

- **Mapping phase**
  - Each task is mapped to the machine giving the earliest completion time

**Heterogeneous Grid**

**Execution times**

<table>
<thead>
<tr>
<th></th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>B</td>
<td>9</td>
<td>13</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>7</td>
<td>10</td>
<td>10</td>
<td>9</td>
</tr>
</tbody>
</table>

**Data transfer times**

<table>
<thead>
<tr>
<th></th>
<th>R1→R2</th>
<th>R1→R3</th>
<th>R2→R3</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>A→B</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>A→C</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>B→D</td>
<td>7</td>
<td>4</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>C→D</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>
Final Schedule: Gantt Chart

- **Task A**
  - \( R1: 5, R2: 8, R3: 8 \)

- **Task B**
  - \( R1: 14, R2: 24, R3: 20 \)

- **Task C**
  - \( R1: 17, R2: 13, R3: 12 \)

- **Task D**
  - \( R1: 21, R2: 31, R3: 28 \)
Bibliography

- **GGF Grid Scheduling Architecture**
  - [https://forge.gridforum.org/projects/gsa-rq](https://forge.gridforum.org/projects/gsa-rq)


- Rizos Sakellariou, Henan Zhao. A Hybrid Heuristic for DAG Scheduling on Heterogeneous Systems

  - 8th Heterogeneous Computing Workshop, IEEE Computer Society Press, 1999

- Yu-Kwong Kwok, Ishfaq Ahmad, Static Scheduling Algorithms for Allocating Directed Task Graphs
  - ACM Computing Surveys, Vol. 31, No. 4, December 1999