ERCIM
European Research Consortium for Informatics and Mathematics

European and Chinese Cooperation on Grid

ECHOGRID
EUROPEAN AND CHINESE COOPERATION ON GRID

Advanced Services for Scientific Workflows

University of Innsbruck, Institute of Computer Science, Austria

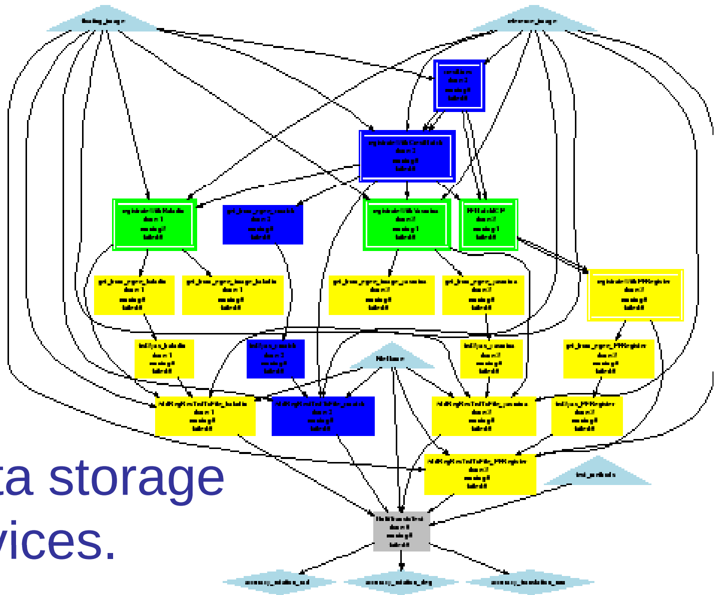
Thomas Fahringer

tf@dps.uibk.ac.at

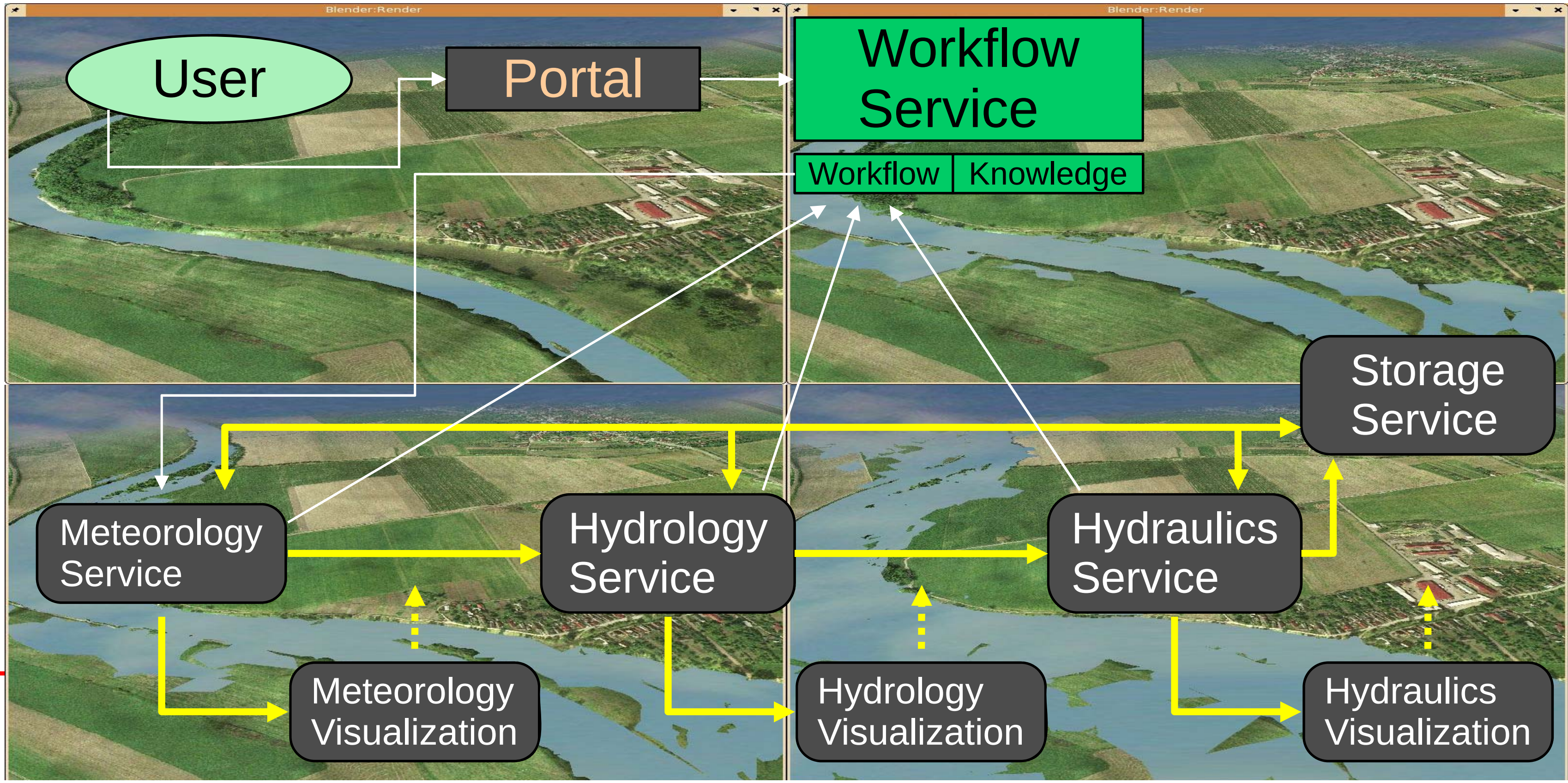
Outline

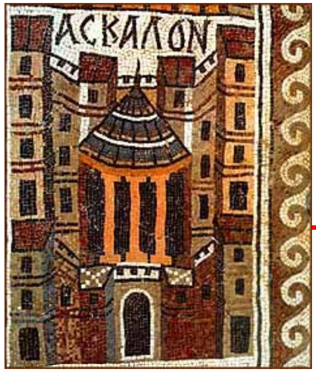
- Grid workflows
- Askalon
  - Application development and runtime environment for scientific Grid workflows
  - Advanced data flow support
  - Semi-automatic resource management
- From scientific to industry Grid applications
  - Online-games
- Summary

Simulate Flooding of the Danube with the Grid



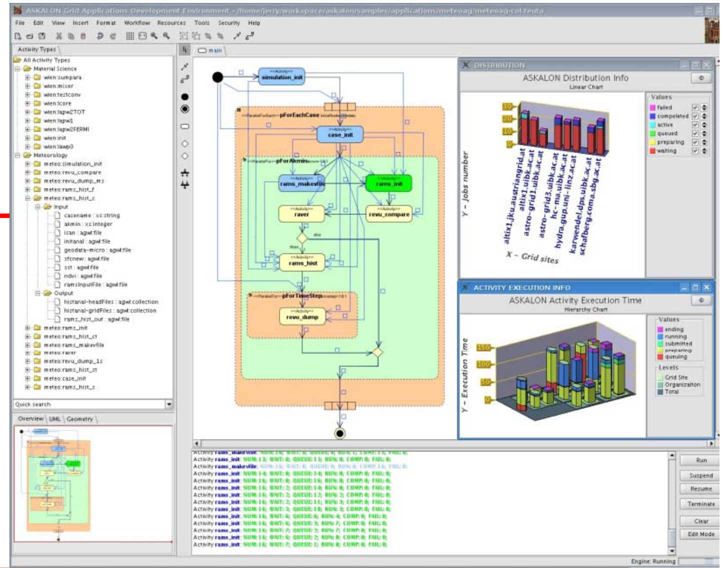
Applications are complex and dynamically constructed from services. Different organisations cooperate to predict the flooding behavior of the Danube by using Grid sensors, computing and data storage resources as well as modeling and simulation services.





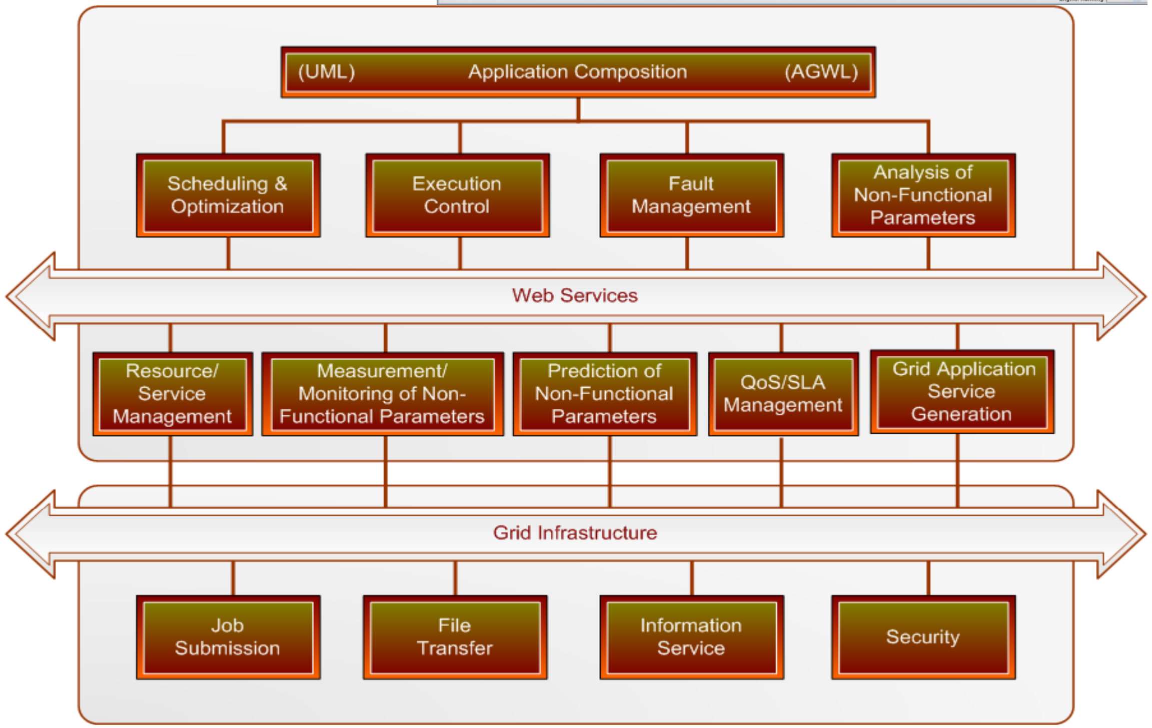
ASKALON

Application Development and Runtime Environment for the Grid

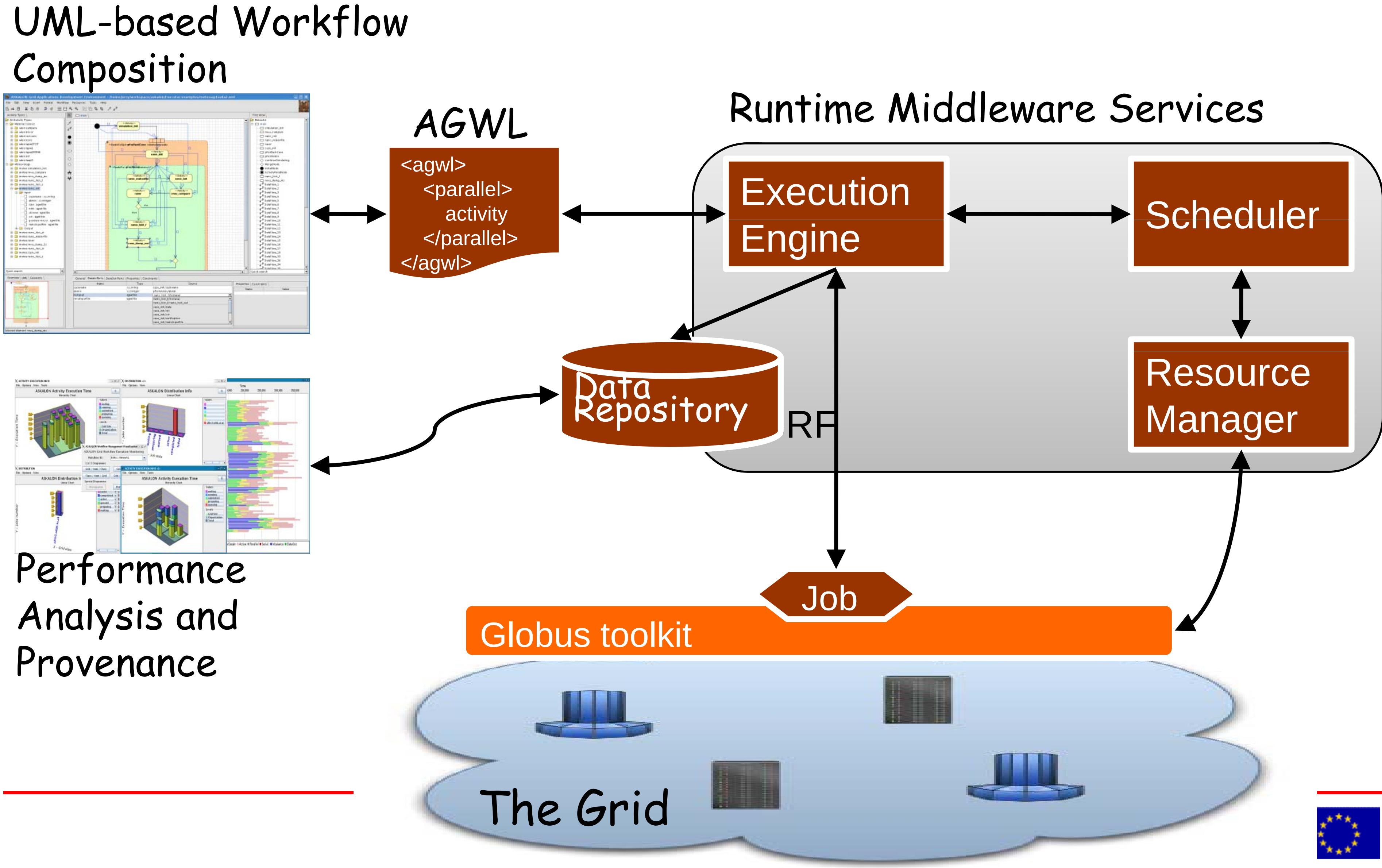


Goal: simple, efficient, effective application development for the Grid

- Invisible Grid
- Application Modeling (UML) and programming at a high level of abstraction (AGWL)
- Semantics technologies
- Semi-automatic deployment
- SOA-based runtime environment with stateful services
- Measurement, analysis and optimization of performance, costs and reliability

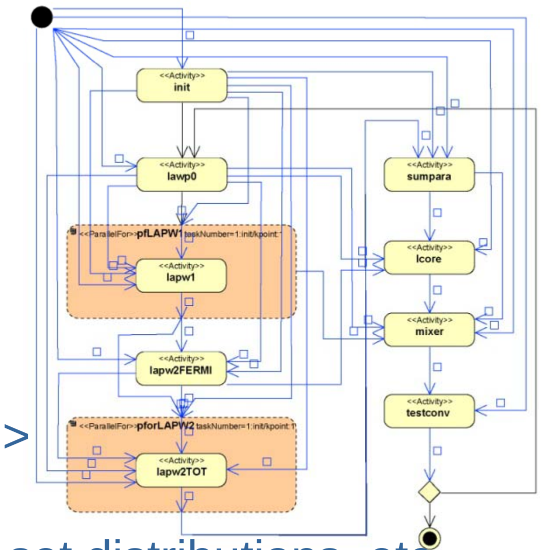


ASKALON Grid Application Composition and Runtime Environment



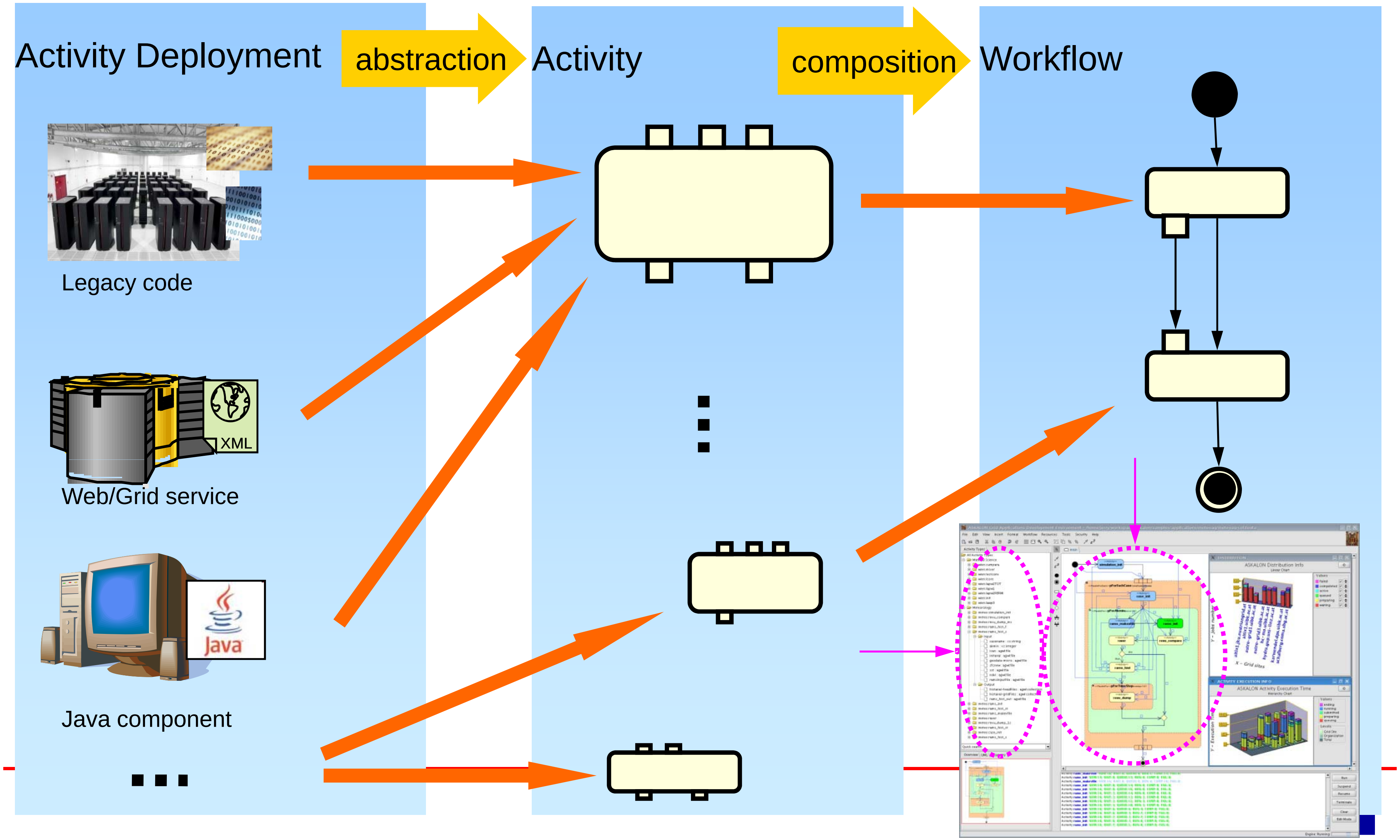
ASKALON: Abstract Grid Workflow Language (AGWL)

- Atomic activities
  - abstract from the real implementation, e.g. Web services, legacy applications
  - Sequential constructs: <sequence>
  - Conditional constructs: <if>, <switch>
- Basic compound activities
  - Loop constructs: <while>, <dowhile>, <for>, <forEach>
  - Directed Acyclic Graph constructs: <dag>
- Advanced compound activities
  - Parallel section constructs: <parallel>
  - Parallel loop constructs: <parallelFor>, <parallelForEach>



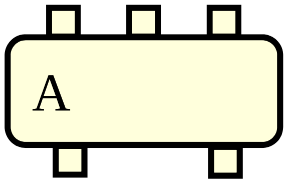
- Data flow constructs
  - dataIn/dataOut ports, collections, data repositories, data set distributions, etc.
- Properties
  - provide hints about the behavior of activities
  - Predicted I/O data size, computational complexity, non-functional parameters
- Constraints
  - Optimization metric (e.g. performance, cost, fault tolerance)
  - Scheduling constraints (e.g. compute architecture, disk, memory)

ASKALON Workflow Composition



Activity Types

- Logical representation of a group of activity deployments (deployed in the Grid), which
  - Realize the same functionality
  - Have the same input/output data structure



UML representation

```xml
<activity name="A" type="type Of A">
    <dataIns>
        <dataIn name="a" .../>
        <dataIn name="b" .../>
        <dataIn name="c" .../>
    </dataIns>
    <dataOuts>
        <dataOut name="d" .../>
        <dataOut name="e" .../>
    </dataOuts>
</activity>
```

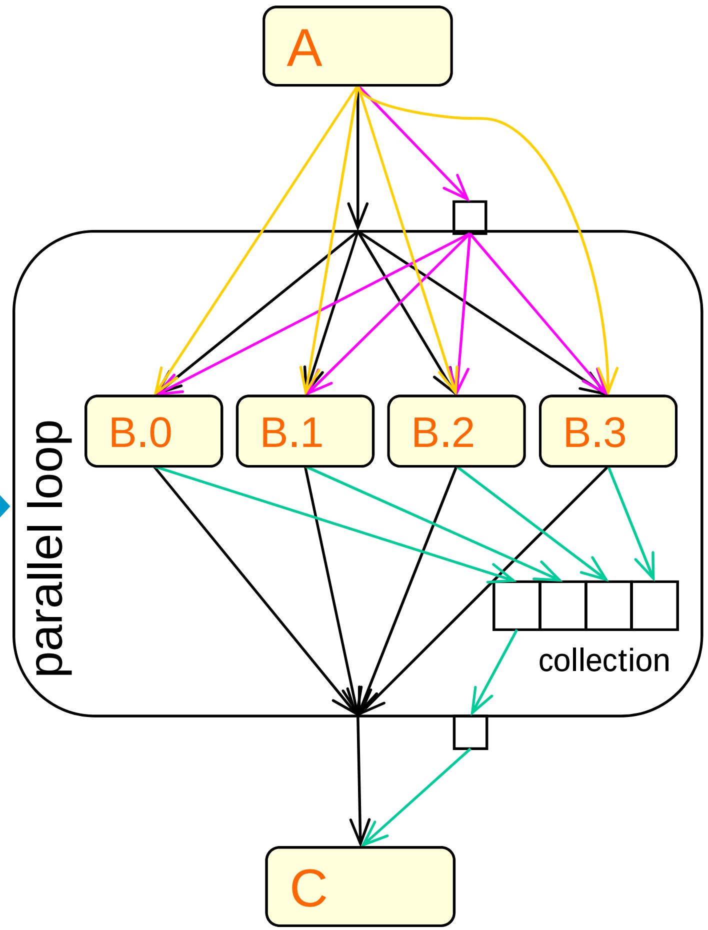
AGWL representation

<parallelFor> Compound Activity

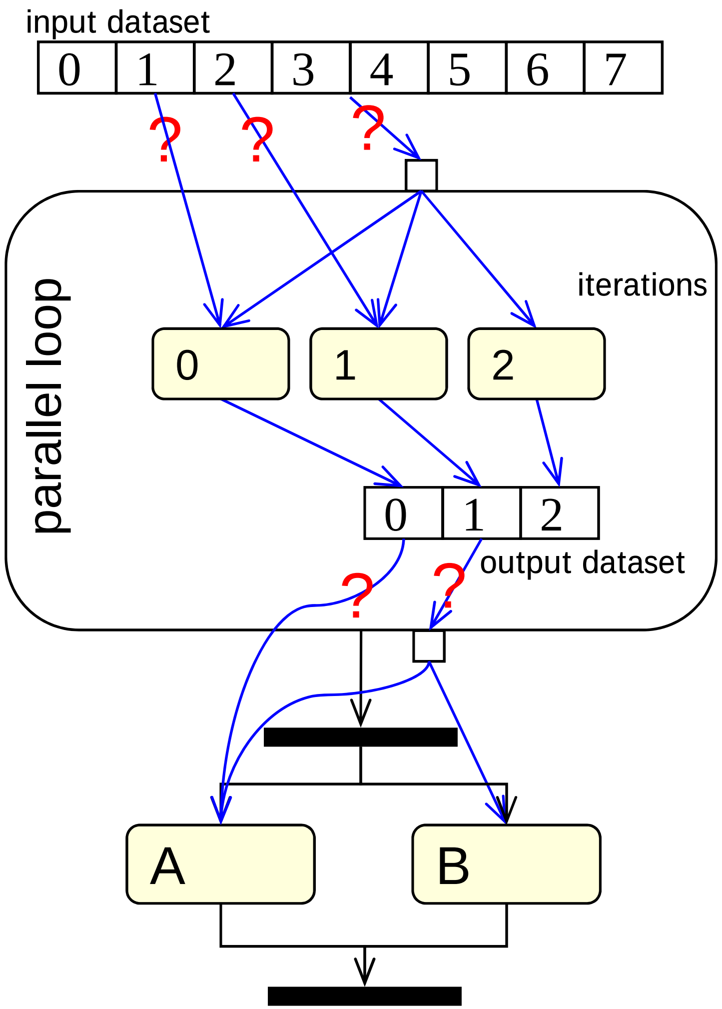
```
<activity name="A" ... />

<parallelFor name="pfor">
  <dataIns>
    <dataIn name="in" source="A/out" >
  </dataIns>
  <loopCounter from="0" to="3" step="1" />
  <loopBody>
    <activity name="B">
      <dataIns … />
      <dataOuts … />
    </activity>
  </loopBody>
  <dataOuts ... />
</parallelFor>

<activity name="C" ... />
```



Data Flow Problems



Problem:
How to distribute datasets onto loop iterations

Status Quo:
Distribute entire input datasets onto each iteration

Better:
Enable distribution of specific dataset parts onto specific iterations, as well as specification of entire datasets onto each iteration

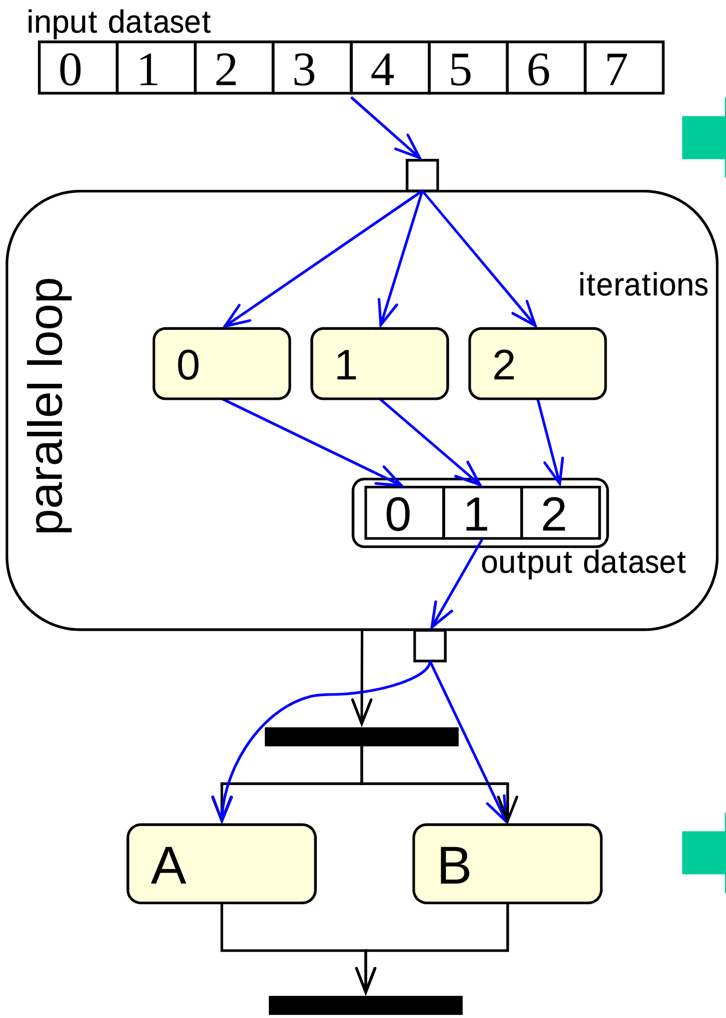
Problem:
How to access output dataset parts?

Status Quo:
Specify the entire aggregated output dataset

Better:
Enable specification of individual data elements (produced by individual iterations) of output datasets, as well as specification of entire output datasets

Flexible dataset-oriented data flow mechanisms to optimize communication

AGWL Approach



AGWL constraint: distribution

```
<dataOut name="c" type="agwl:collection"/>
...
<parallelFor name="PFor" ...>
  ...
  <dataIn name="in" type="agwl:collection"
source=".../c">
    <constraints>
      <constraint name="distribution" value="BLOCK(3)"/>
    </constraints>
  </dataIn>
</parallelFor>
```

specifies that blocks of 2 data elements of the input dataset are assigned to every different iteration:

data 0, 1, 2: iteration 0
data 3, 4, 5: iteration 1
data 6, 7 : iteration 2

AGWL constraint: element-index

Activity A: <constraint name="element-index" value="0,2"/>
to specify the data elements with index 0,2

Activity B: no constraint "element-index"
to specify the entire dataset

Data Collection Distributions

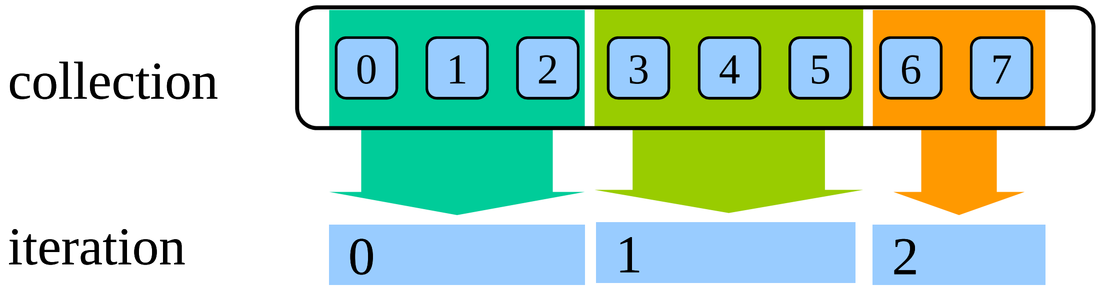
- Distribution collections onto loop iterations: ***distribution***
  - BLOCK, **BLOCK(S)**, BLOCK(S,L), REPLICA(S)

$$\delta(i) = \left\{ \left\lfloor \frac{i}{S} \right\rfloor \,\middle|\, 0 \le i < |C| \;\wedge\; S \ge \left\lceil \frac{|C|}{|I|} \right\rceil \right\}$$

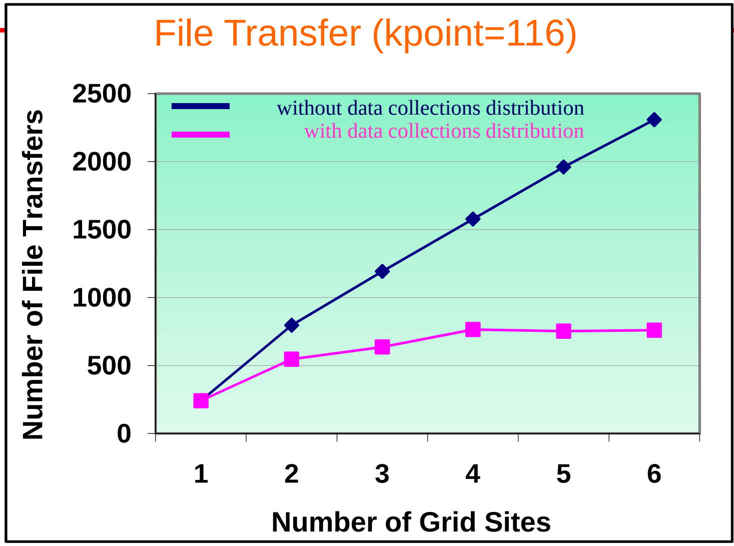
i	data element index
δ(i)	distribution function
|*C*|	Collection size
|*I*|	iteration size
S	block size

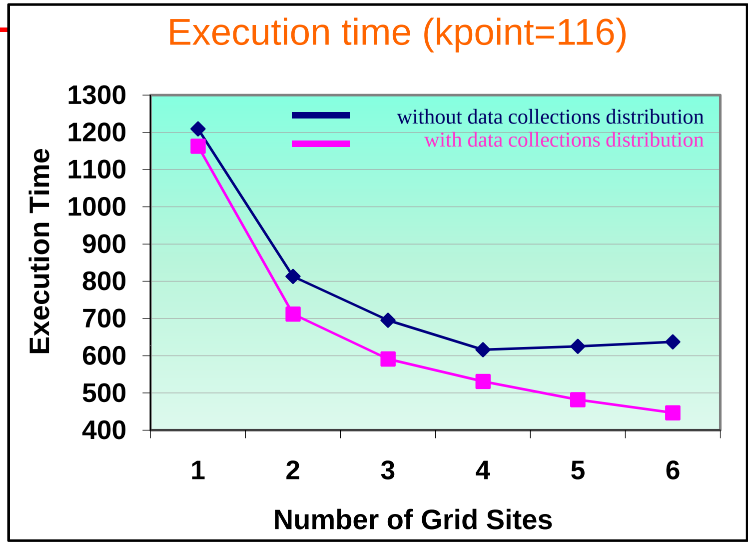
> Example:

distribution="BLOCK(3)" $|C| = 8$, $|I| = 3$



Performance Results of Wien2K





kpoint=116

File transfer:

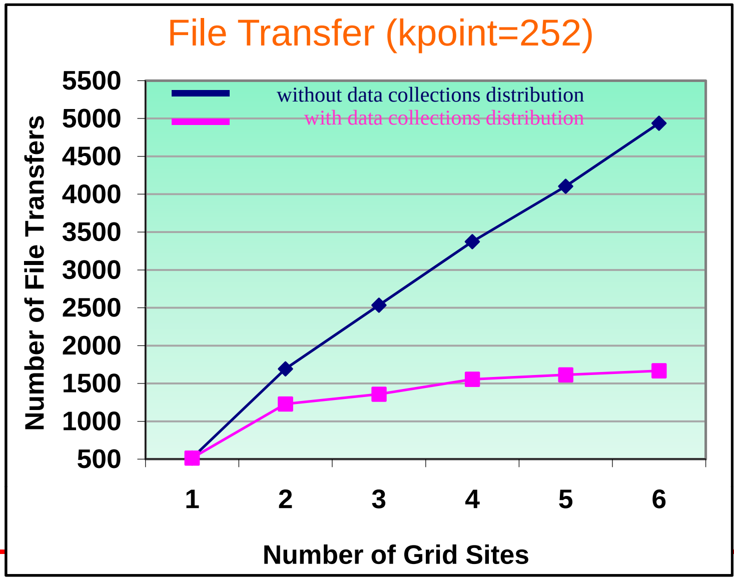
67% reduced

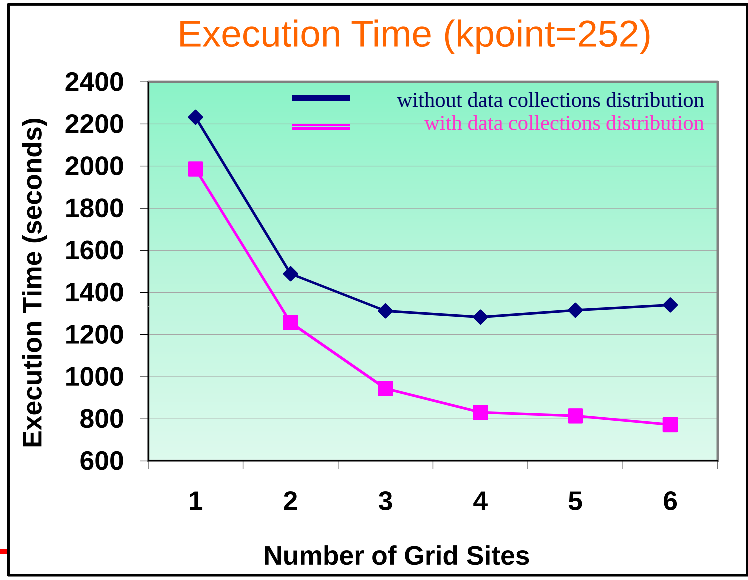
Execution time:

30% reduced

Speedup (max):

1.96 vs. 2.9





kpoint=252

File transfer:

68% reduced

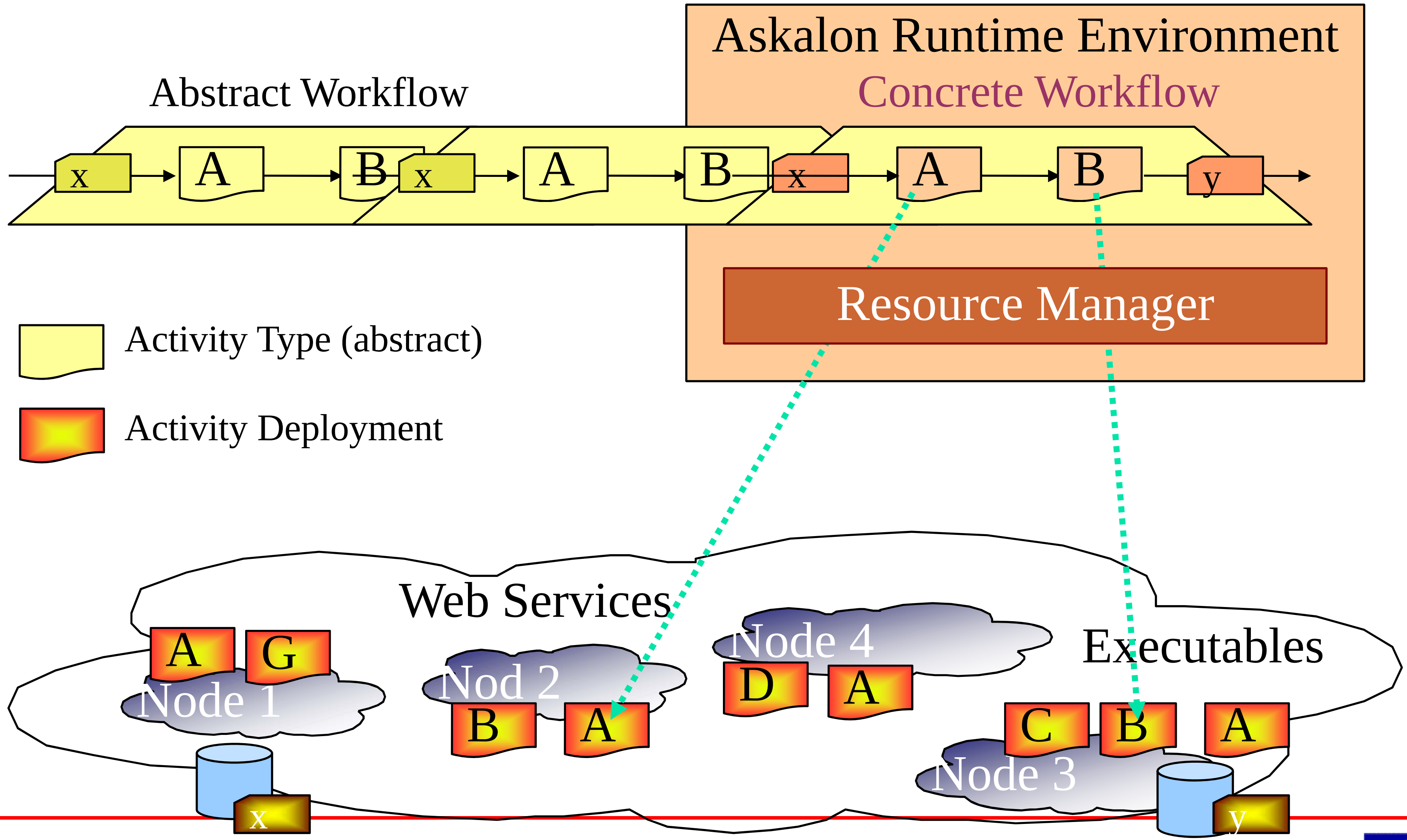
Execution time:

42% reduced

Speedup (max):

1.74 vs. 2.58

Dynamic Bindings of Workflow
Abstract - Concrete



Build File

Prepare for Installation

Setup Environment variables,
create directories etc

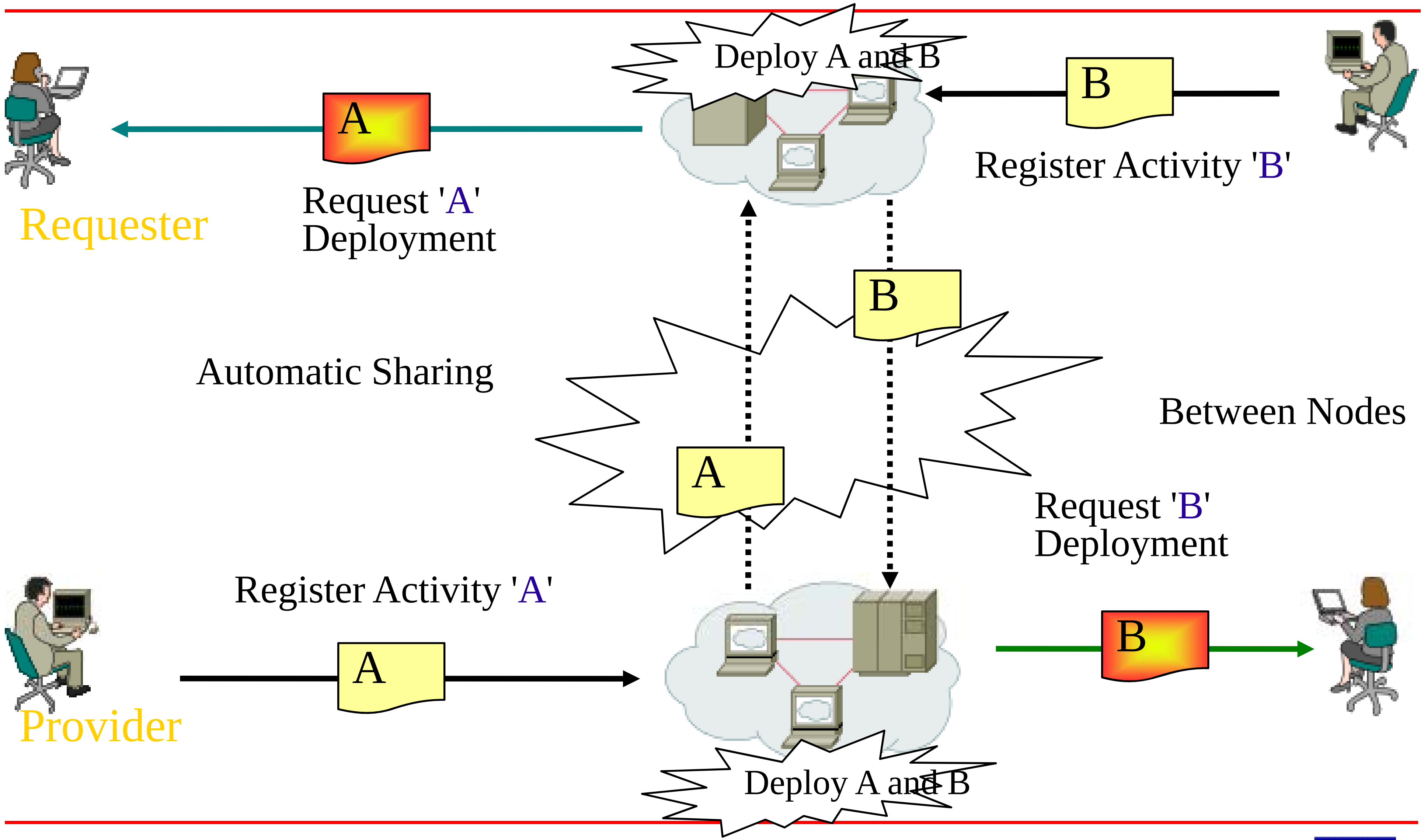
Download Installable File

Deploy

Undeploy

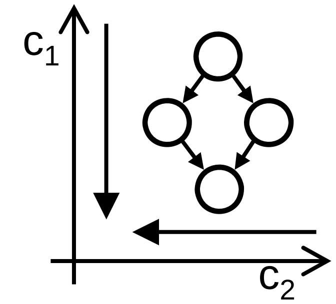
```
<Build baseDir="$DEPLOYMENT_DIR" xmlns=http://bu
  defaultTask="Deploy" name="jpovray">
  <Step name="Init" task="mkdir -p" baseDir="$DEI
      <Env name="ACTIVITY_HOME" value="jpovray"/>
      <Env name="ACTIVITY_TARBALL" value="$ACTIVI
      <Env name="SRC_URL" value="http://dps.uibk
      <Env name="ACTIVITY_HOME_PATH"
                          value="$DEPLOYMENT_DIR/$AC
      <Env name="BIN_DIR" value="$ACTIVITY_HOME_P
      <Property name="argument" value="$DEPLOYME
  </Step>
  <Step name="Clean" depends="Init" task="rm -r"
                            baseDir="$DEPLOYMENT_DIR
    <Property name="argument" value="$ACTIVITY_TA
    <Property name="argument" value="$ACTIVITY_HC
  </Step>
  <Step name        ="Download" depends="Clean"
        task        ="$GLOBUS_LOCATION/bin/globus-
        baseDir     ="$DEPLOYMENT_DIR" timeout="80"
    <Property name="source" value="$SRC_URL/$ACT
    <Property name="destination"
              value="file:///$DEPLOYMENT_DIR/$ACT
    <Property name="md5sum" value="ksljddsfdsjdf
  </Step>
  <Step name="Deploy" depends="Download" task="t
                          baseDir="$DEPLOYMENT_DIR"
    <Property name="argument" value="$ACTIVITY_TA
    <Dialog expect="Install as root or normal us
  </Step>
  <Step name="Undeploy" depends="Clean"/>
</Build>
```

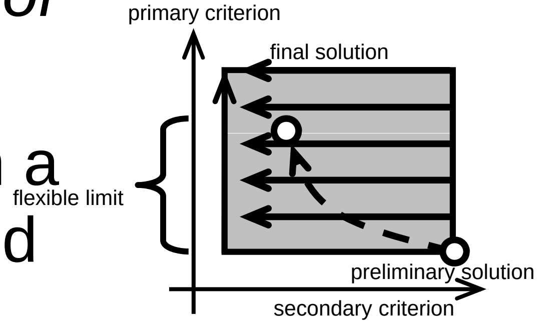
Registration and Discovery

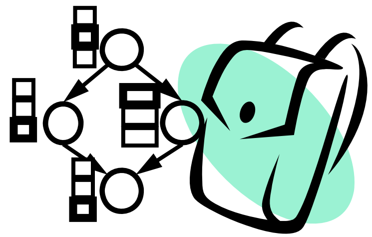


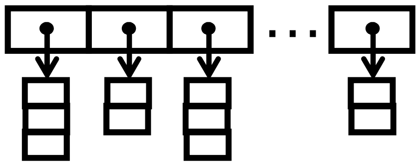
ASKALON
Bi-parameter Scheduling

- *Bi-parameter* scheduling of DAGs
- Generic parameter model based on a novel *taxonomy of criteria*
- Parameter minimization with a *flexible constraint* established for one parameter
- Problem describes as *multiple choice knapsack problem*
- Two-phase optimization based on *dynamic programming*



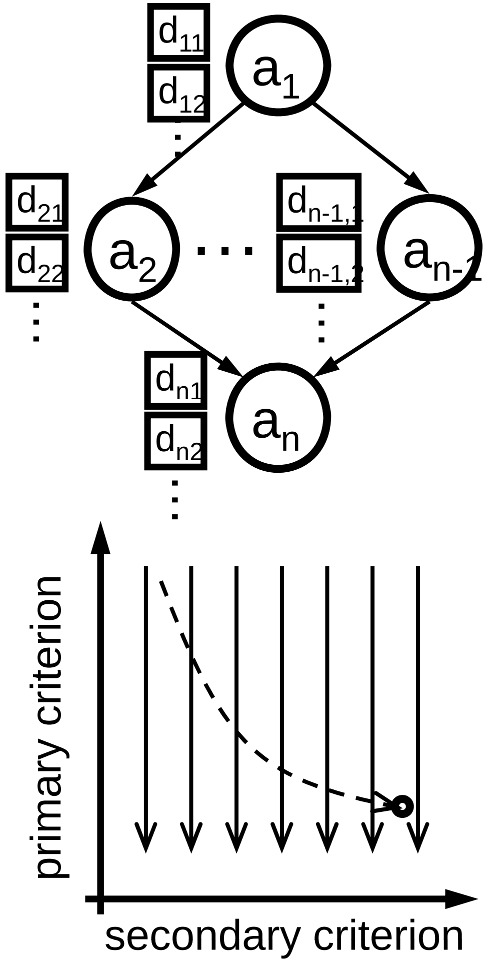






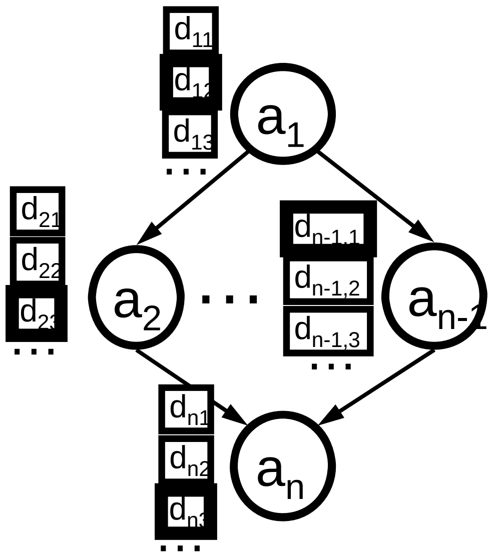
Primary Scheduling

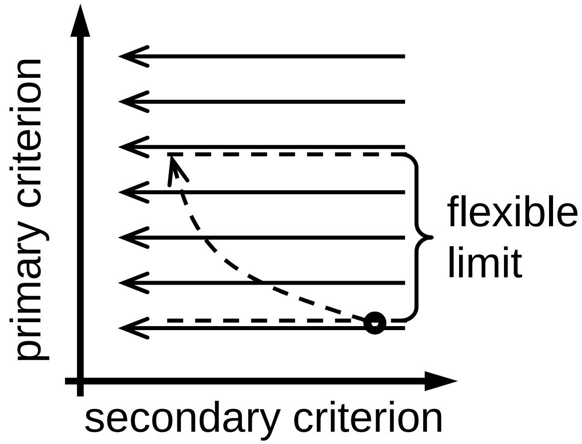
- Goal: optimize the schedule for the primary criterion only
  - NP-complete for intradependent criteria
    - E.g. execution time
  - Trivial for non-intradependent criterion (simple greedy approach)
    - E.g. cost
- Many heuristics for optimization of time
  - HEFT algorithm
- Result: *preliminary solution*



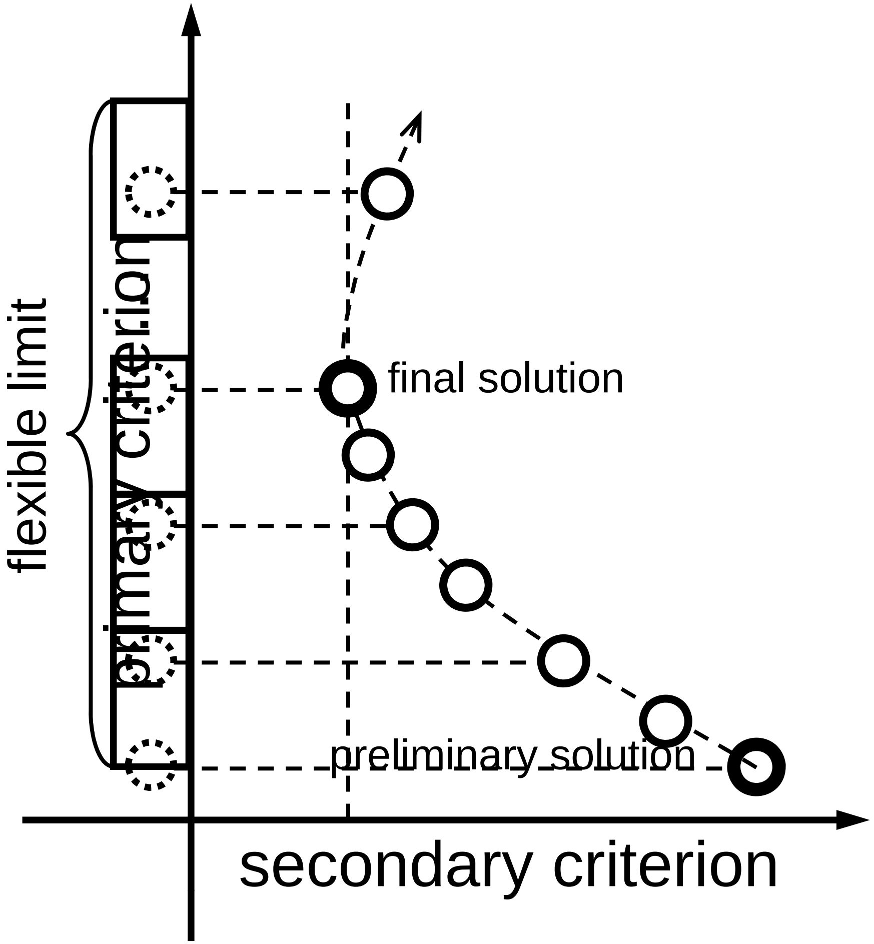
Secondary Scheduling (1/2)

- Goal: modify the preliminary solution, optimizing the secondary criterion
- The primary criterion kept within the **flexible limit**
- The problem modeled as the *multiple choice knapsack problem*
- Solution based on *dynamic programming*





Secondary Scheduling (2/2)

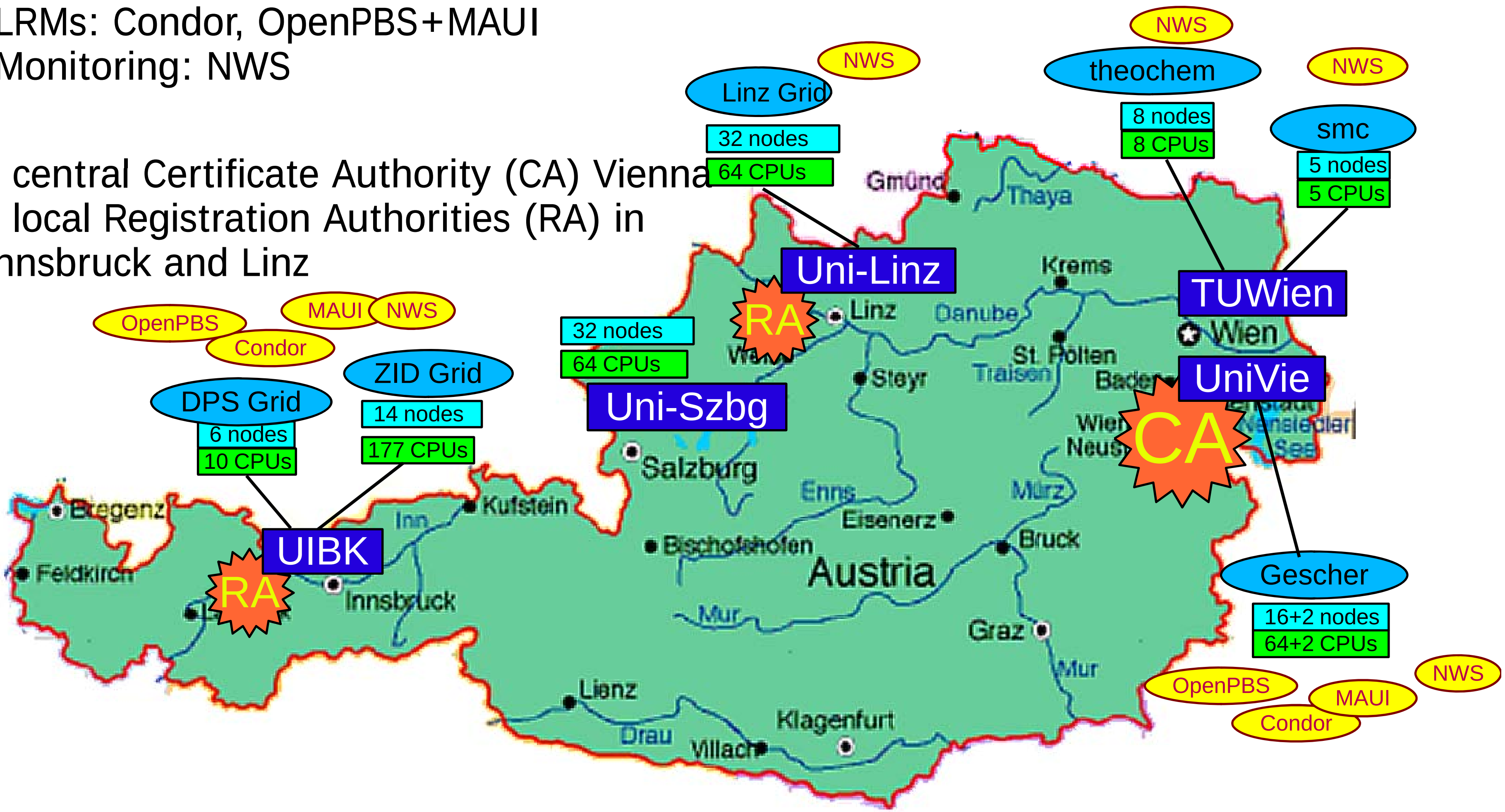


- An auxiliary table stores all *intermediate solutions*.
- Initially, only the preliminary solution.
- Only the *non-dominated* solutions are stored in the table.
- Finally, the *final solution* selected.

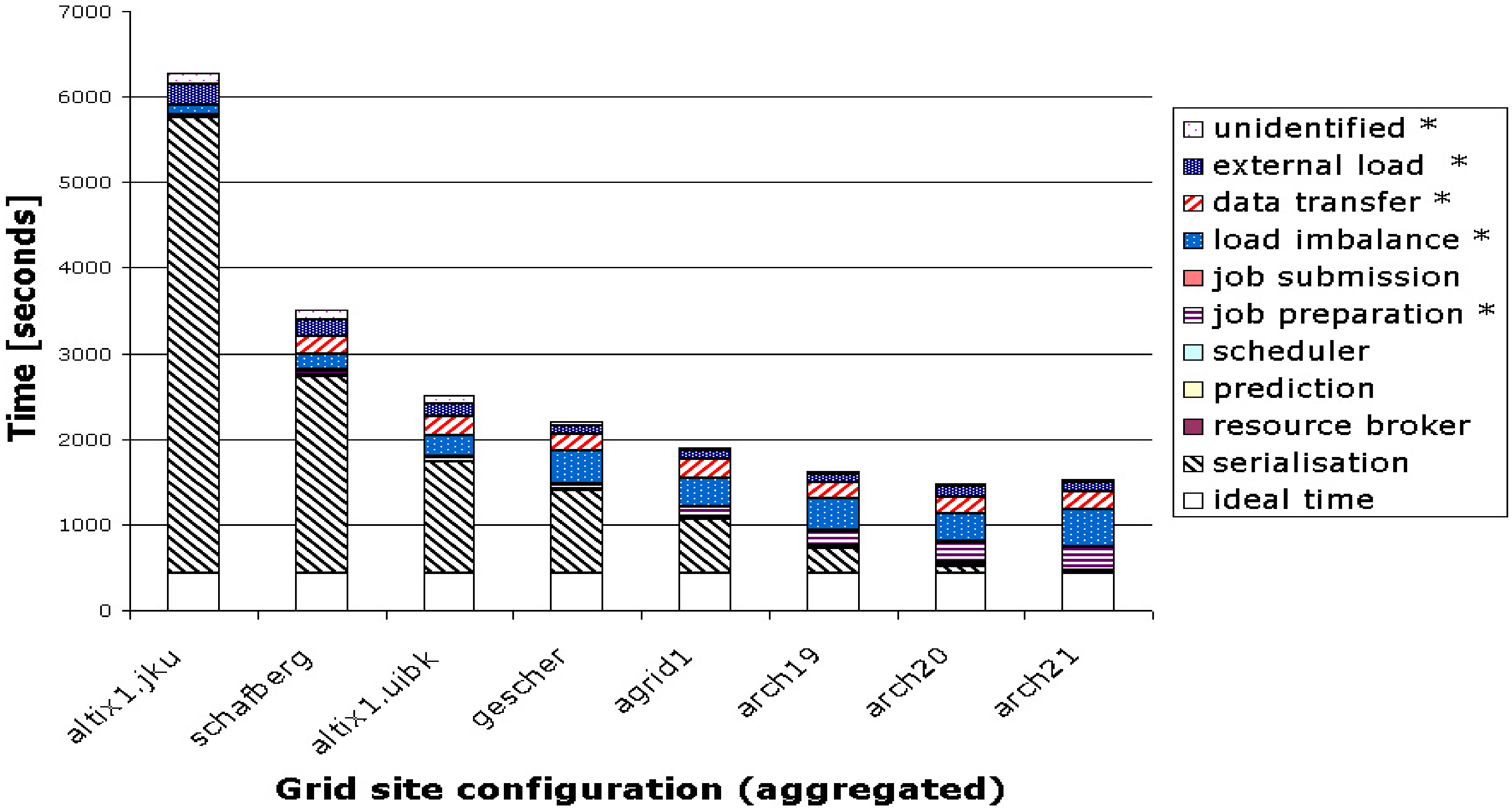
Austrian Grid

Grid environment: Globus Toolkit 2.4
LRMs: Condor, OpenPBS+MAUI
Monitoring: NWS

1 central Certificate Authority (CA) Vienna
2 local Registration Authorities (RA) in
Innsbruck and Linz



Scalability Experiments



Edutain@Grid

www.edutaingrid.eu

EU funded STREP

edutain@grid

A Grid Environment for Interactive Gaming and E-Learning

Lead Partner and Project Details:

Thomas Fahringer, University of Innsbruck, Austria

7 partners in total, approx. 2,5 Mio Euro total funding

Project duration: Sept 2006 – Aug. 2009

University of Münster (Germany), IT Innovation (UK), University of Linz (Austria), Darkworks (France), BMT (UK), Amis (Slovenia)

universität innsbruck

Westfälische Wilhelms-Universität Münster

innovation

BMT

reliably amis

DARKWORKS VIDEO GAME DEVELOPMENT STUDIO

Information Society Technologies

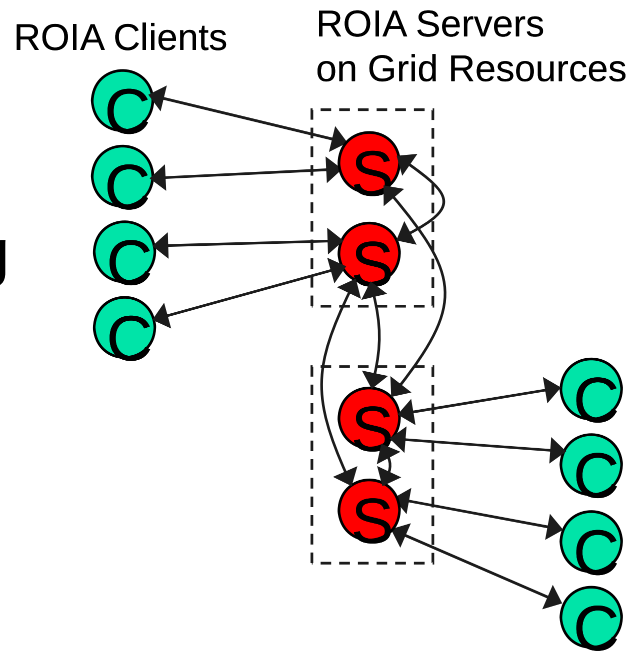
Online-Games:
Killer-Applications for Grids

Huge Market

- Videogame Market: €33 billion in 2007
- €0,7 billion online games revenue in Europe

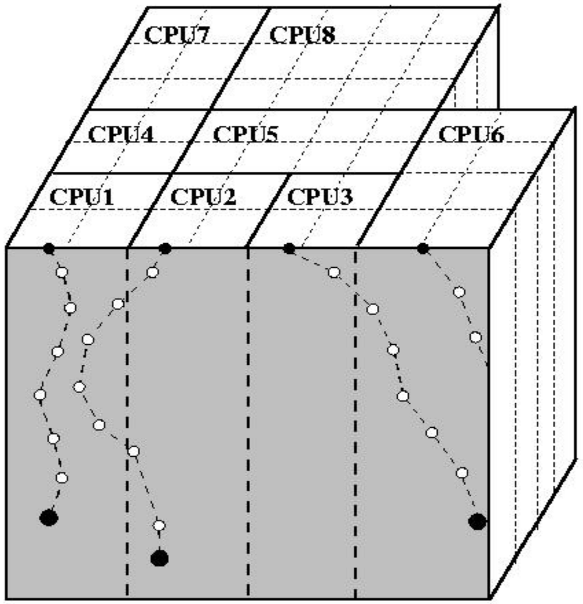
Real-time Interactive Applications (ROIA)

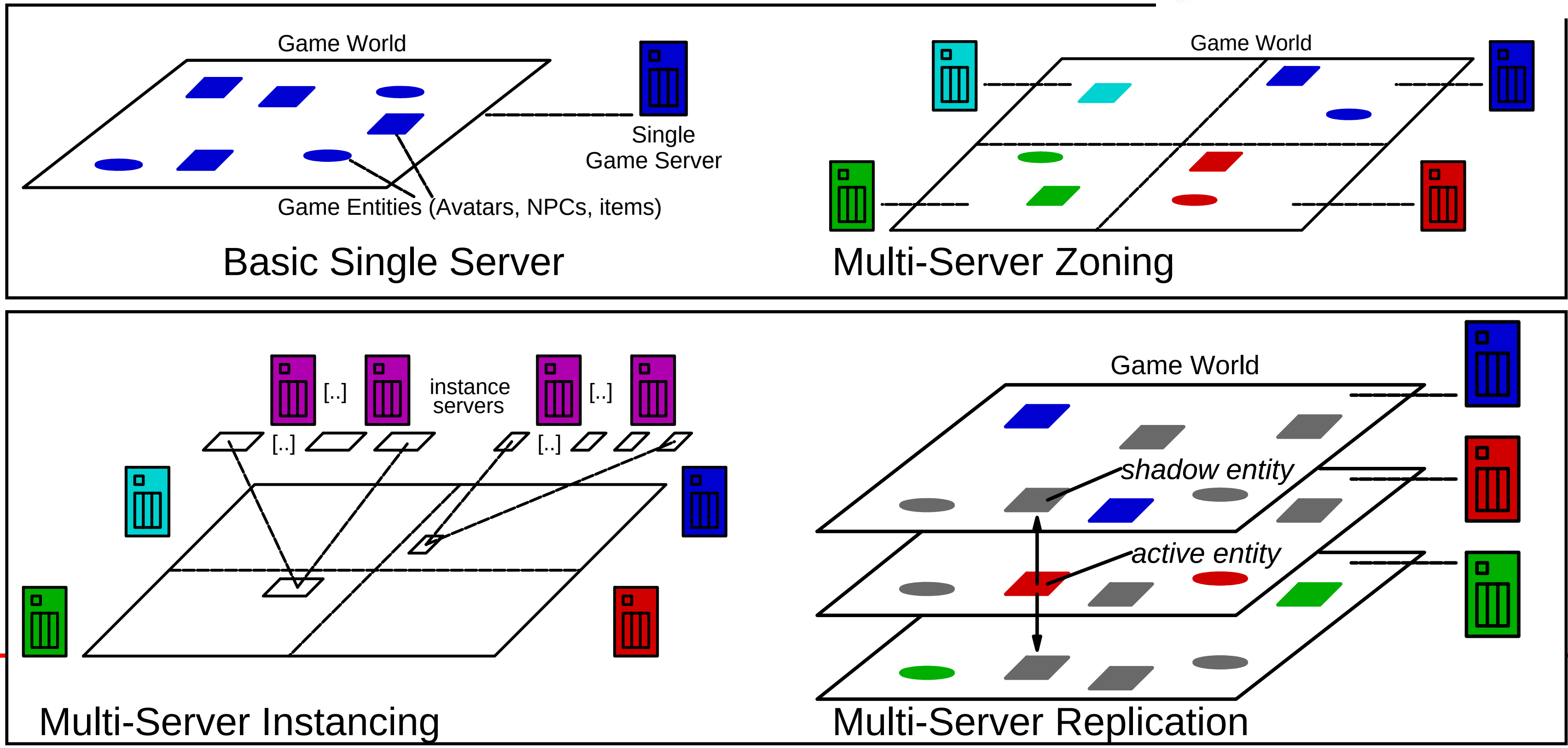
- Mediates and responds in real-time to highly frequent user interactions
  - e.g. 35Hz
- Delivers and maintains well-defined QoS parameters related to the user interactivity
  - E.g. Number of updates per seconds
- Highly dynamic and adaptive to changing user interaction loads
- Ad-hoc user connections, often by using anonymous or different pseudonyms
- Competition-oriented Virtual Organisations



edutain@grid Scalability: Virtual World Parallelization

- Similar to n-body or particle-in-cell problem
- Processing of game avatars entities on the Grid
- Distribution of avatars by a built-in framework:
- *zones, instancing* and *replication* will be supported





Summary

- **Current Workflow Systems**
  - Application developer deals with Grid details
  - Static binding from workflow to code deployments
  - Many limitations of workflow languages
  - Optimization for runtime only
- **Invisible Grid:**
  - Abstract from Grid details
  - Modeling versus coding
  - Model applications at high level of abstractions
  - Dynamic binding of application model to implementation
  - Semi-automatic resource managment
  - Data flow optimization
  - Multi-parameter optimization for varierty of QoS parameters
- **From scientific applications to industry applications**
- **More information: www.askalon.org**