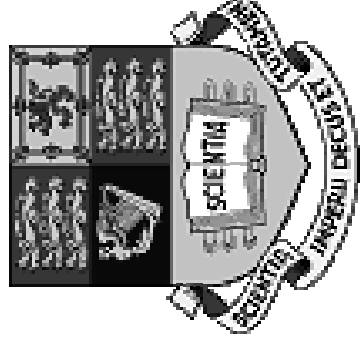


Computational Communities: A Marketplace for Federated Resources

Steven Newhouse



Parallel Software Group
Imperial College Parallel Computing Centre
Department of Computing
Imperial College, London
icpc-sw@doc.ic.ac.uk

Outline

- Motivation for Grids
- Jini Middleware
- Resource Utilisation
- Application Meta -data
- Experiments

Motivations for Grid(s)

- Optimise Use of Resources
 - Wider Access
 - Better Utilisation of Resources (Throughput)
- Connect Facilities
 - Linked Instruments/Storage/Computation
 - Coupled Computing (Meta -computing - High Performance)
- High-level Use (e -science)
 - Ease of Use for Domain Specialists
 - Support Scientific Collaboration and Knowledge Management (Connect Communities)

A More General Model

Computational Communities

Share and combine computational resources (software and hardware) in an open, transparent and optimal manner

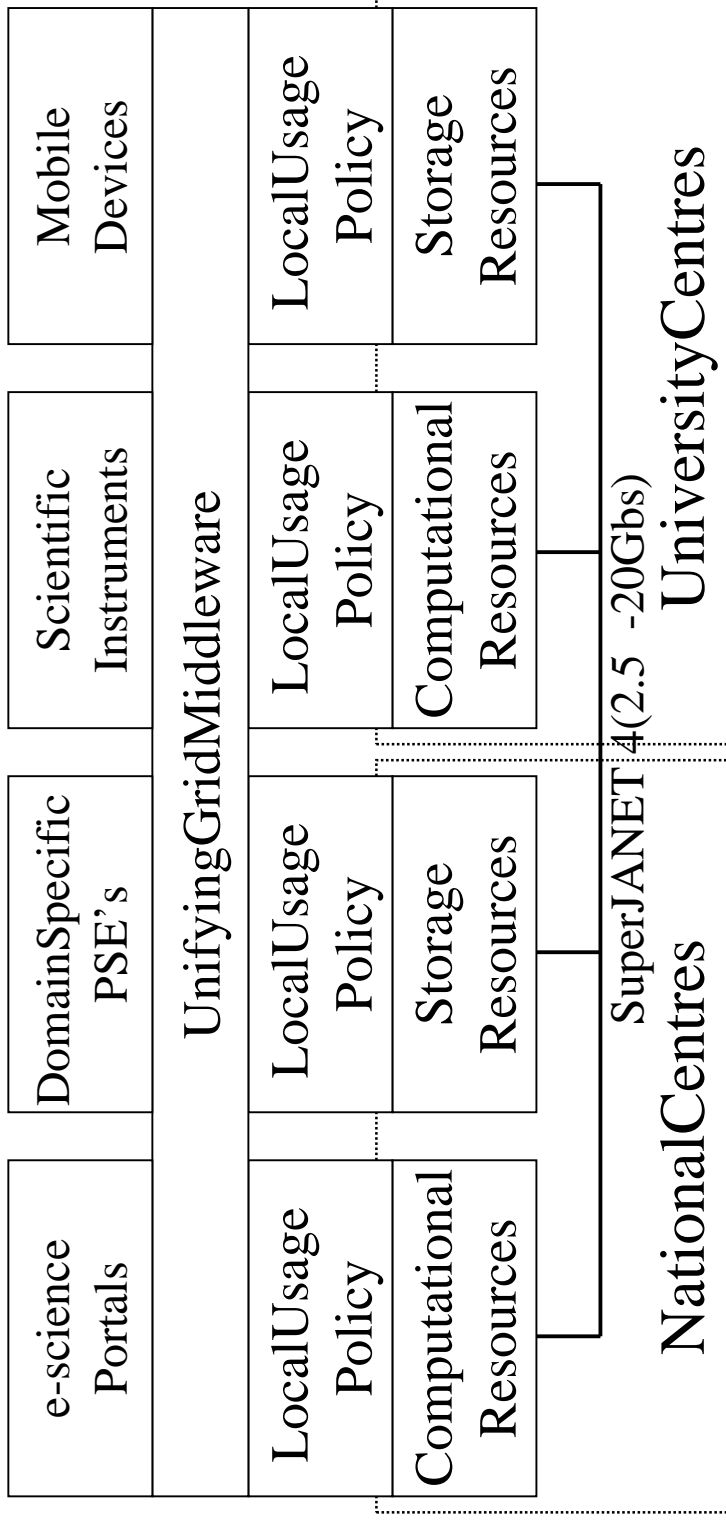
Need information about:

- the applications
- the resources
- the users

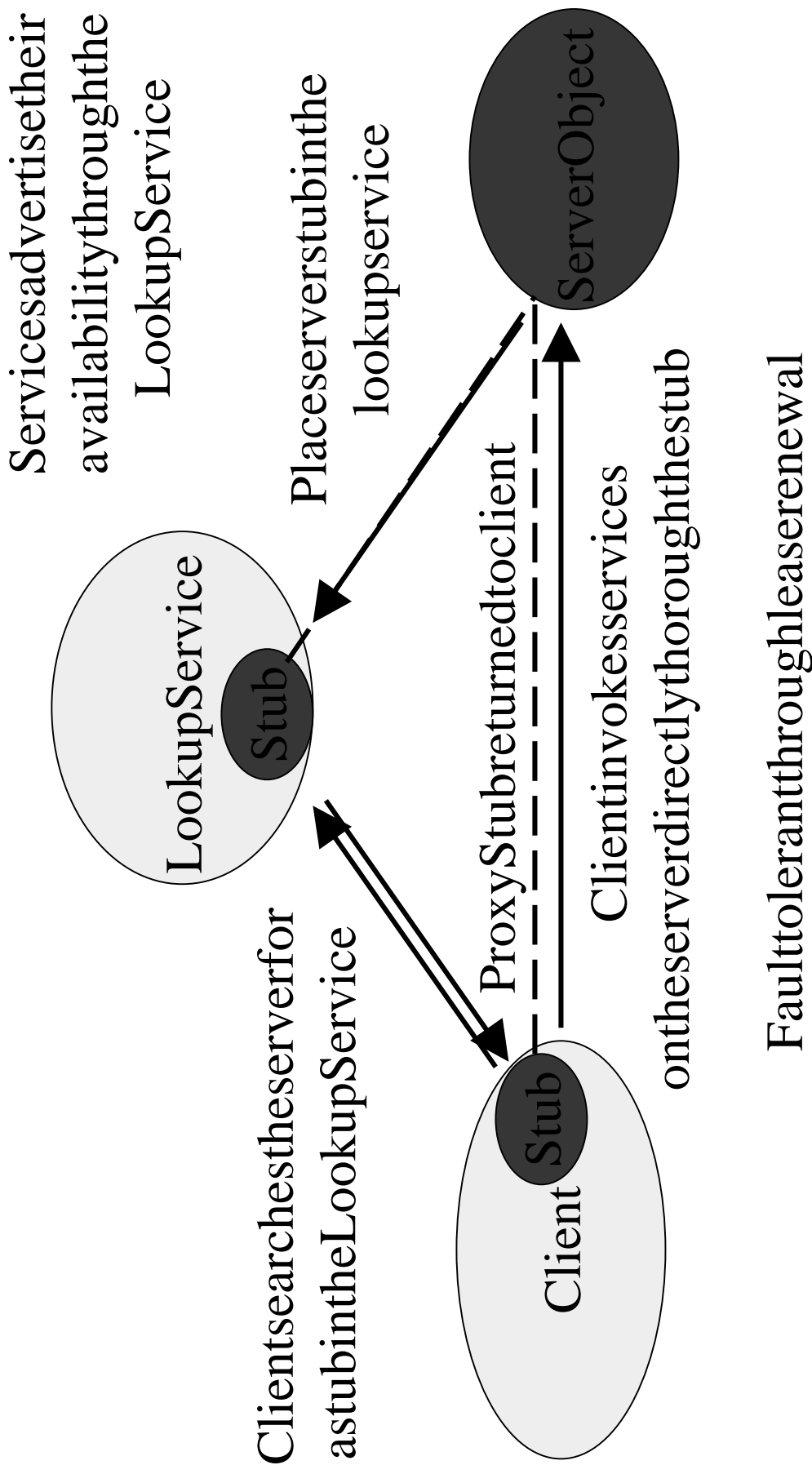
Federation of Grid Resources

- Heterogeneous computational, storage, networking and software resources
- Administrative domains must retain local autonomy
- Fault tolerant decentralised architecture
- Existing systems: Globus, Legion

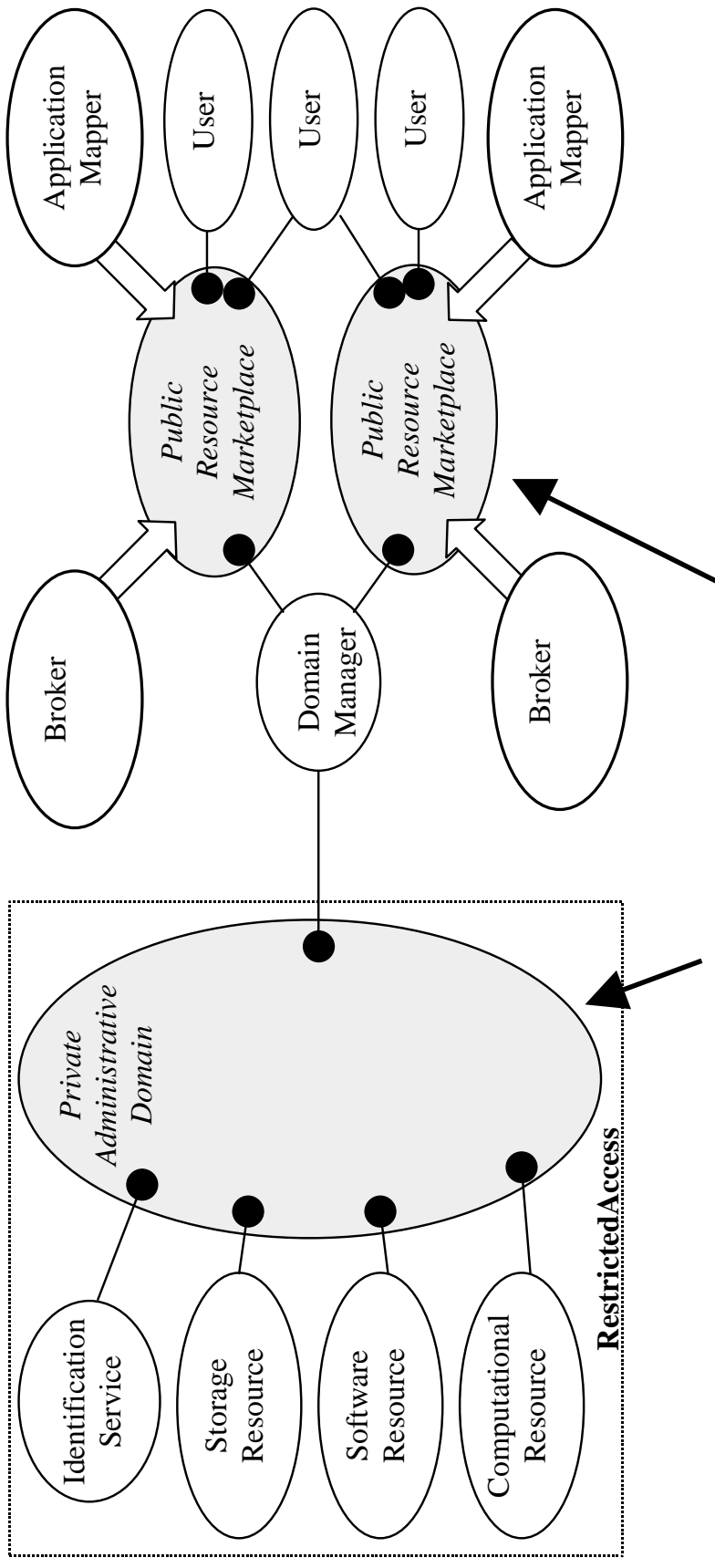
Federating Shared Resources



Jini Architecture

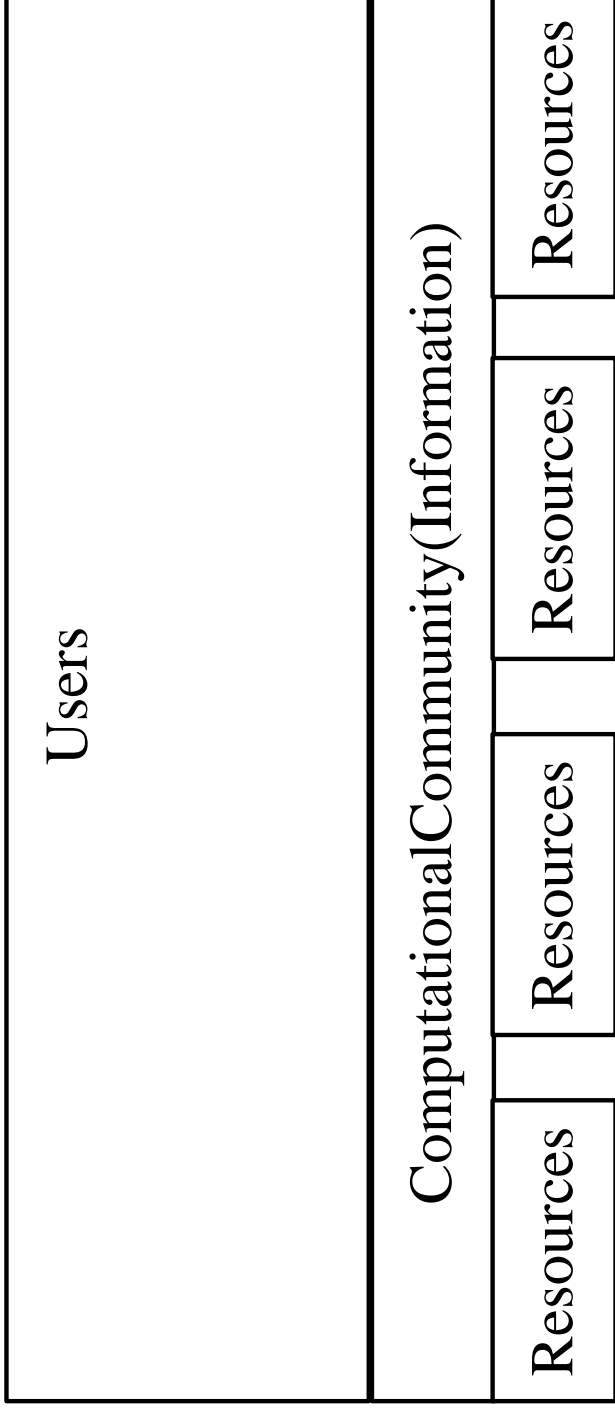


Federating Grid Resources with Jini

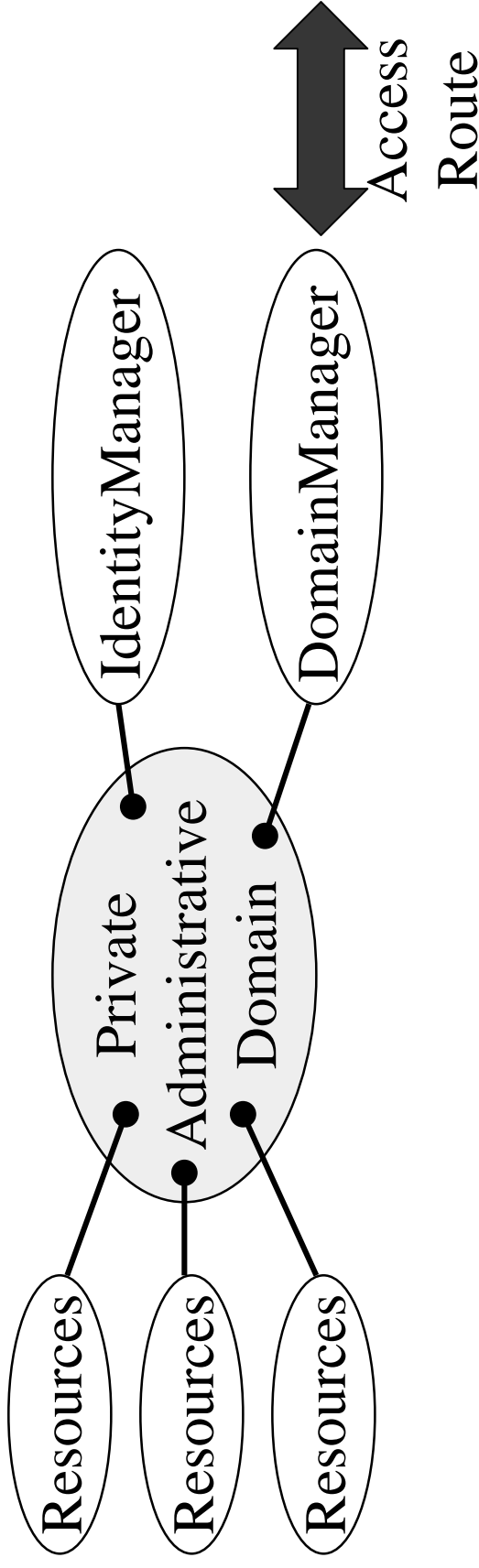


Jini LookUpServers

Building a Grid Environment 1



Jini GridMiddleware

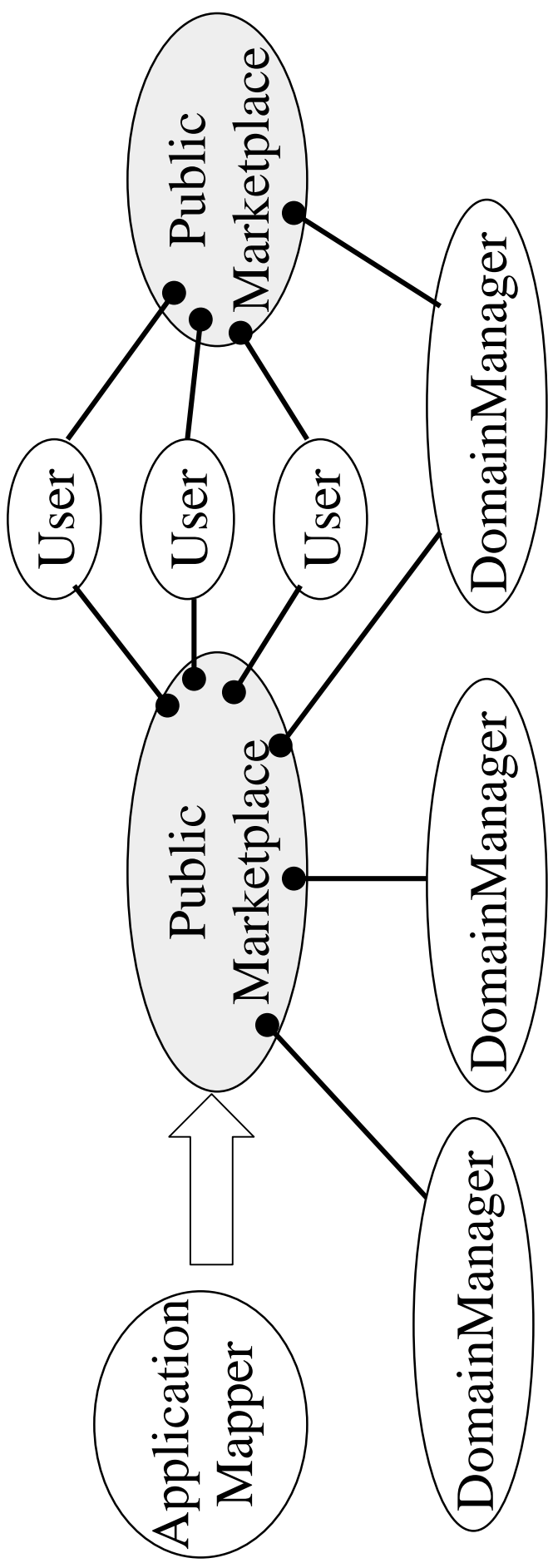


- The resources (computational, software etc.) advertise their capabilities to the private domain
- The Domain Manager imposes local usage requirements and access and publicly advertises its resources
- The Identity Manager authenticates the domains, groups and individuals in the grid ('trusted service')

Trusting Other Domains & Users

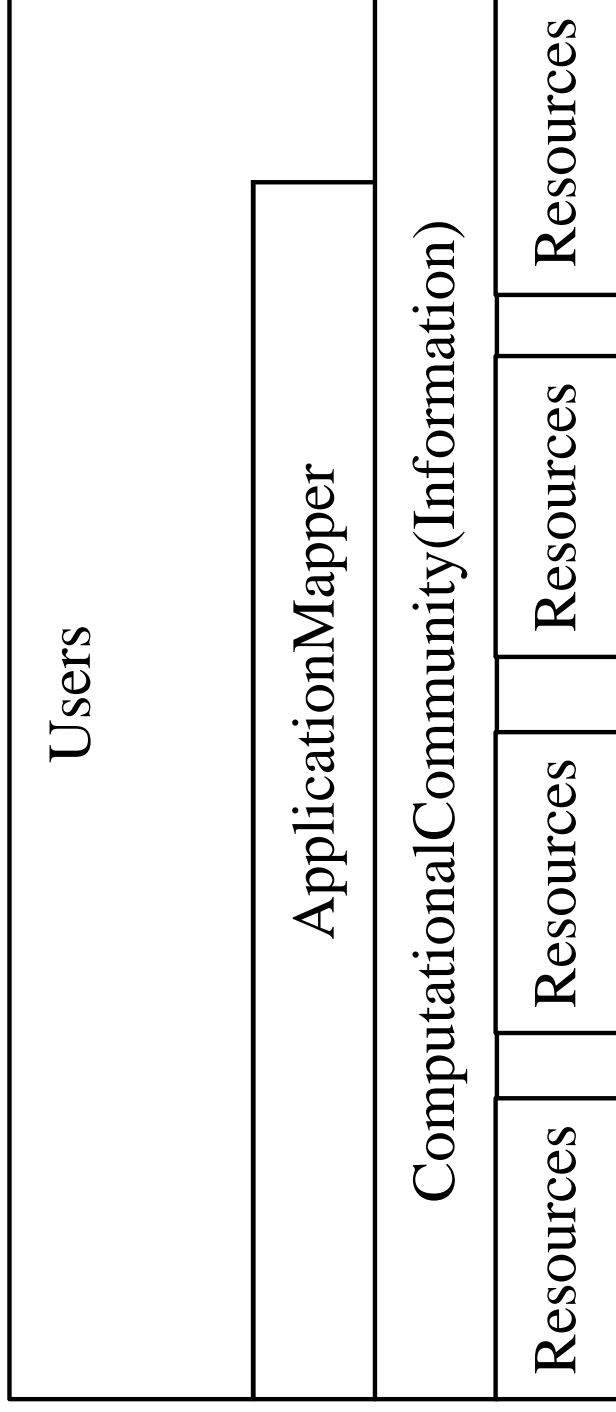
- Recognises three entities - individuals, groups and domains
- Entities verified through public key architecture with authorising Identity Manager
- Locally managed access control list determines which entities have access to local resources
- Non-local users can be mapped to individual, single or guest accounts (site dependent policy)

Federating Domains



- Marketplace allows Users and Domain Manager to advertise their requirements and capabilities
- The 'Application Mapper' finds the 'best' match between jobs and resources

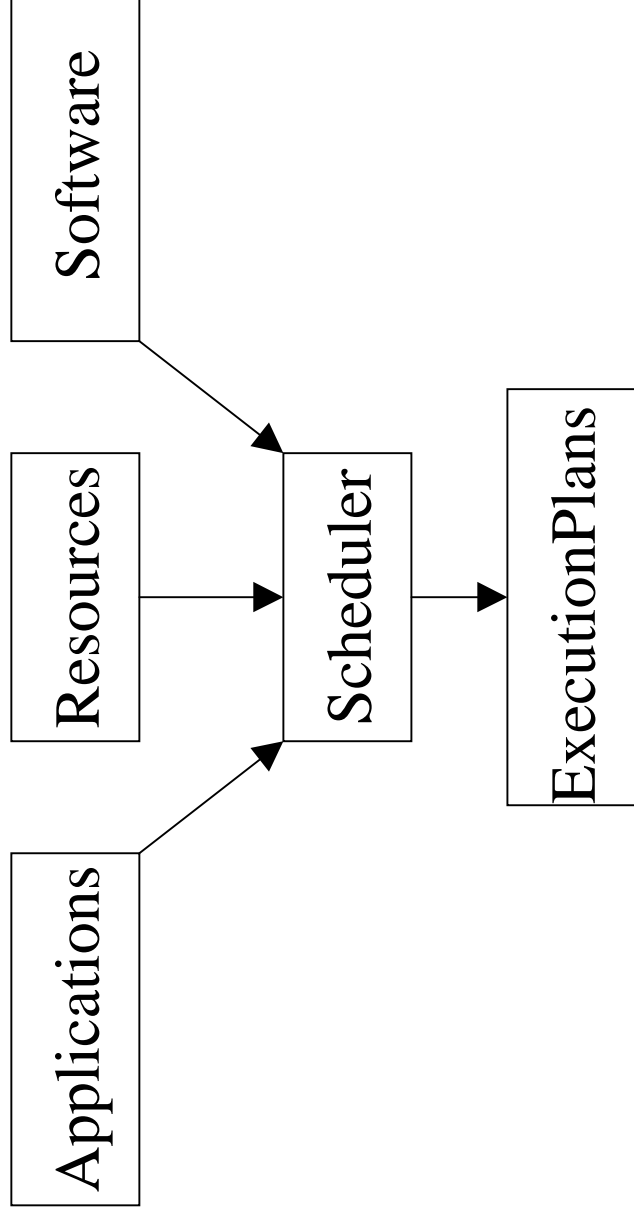
Building a Grid Environment 2



Scheduler Architecture

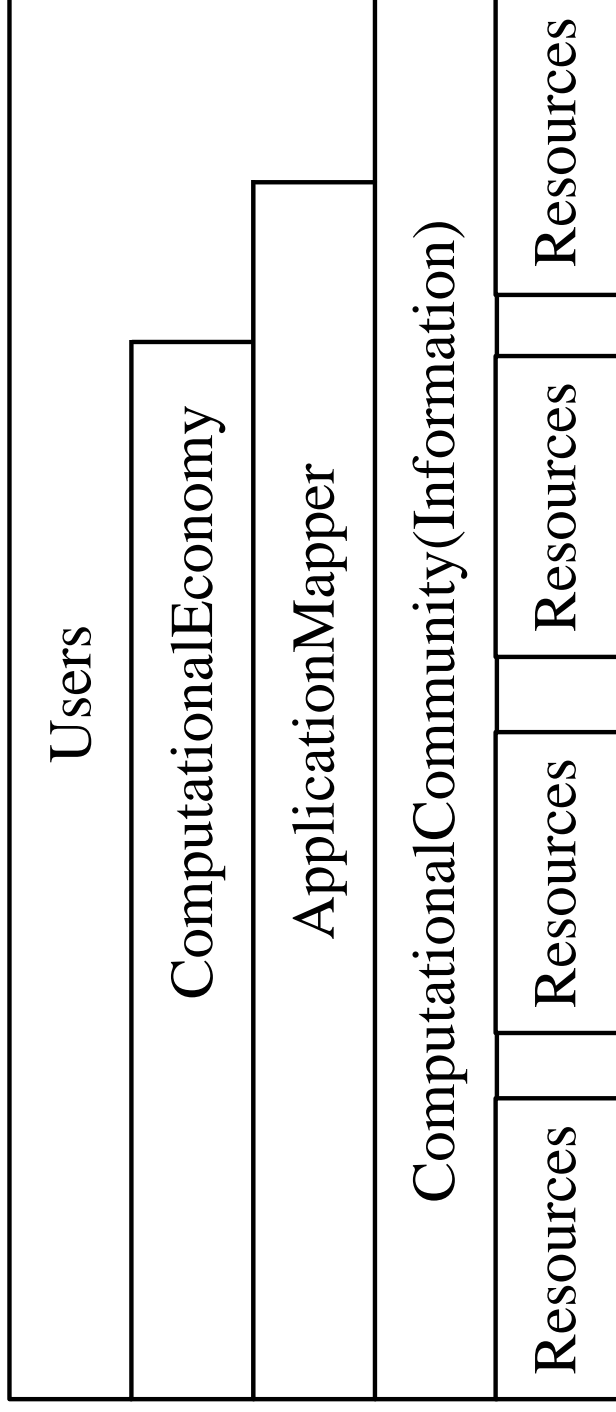
- Maximise throughput over multiple resources & domains
- Scheduler independent of any particular domain
- Users and resources can advertise in multiple marketplaces
- The 'best' scheduler will produce the 'best' marketplace (e.g. cost, execution time, etc)
- Clear separation between:
 - user requirements
 - resource capabilities
 - application execution

Multi-User Scheduling



Need to ensure that the execution plans are globally optimal

Building a Grid Environment 3



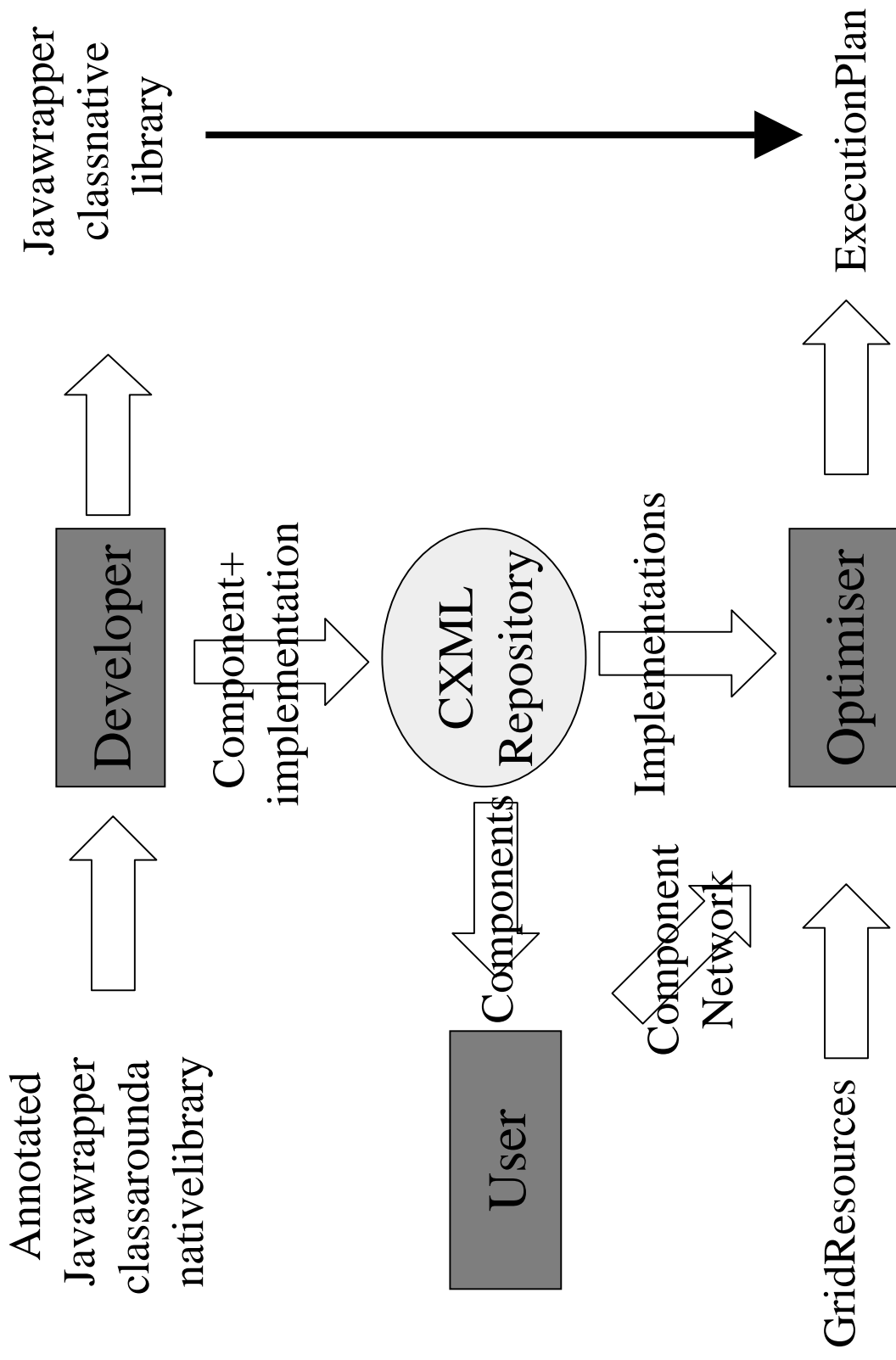
Scheduler Functionality

- Resource costing
 - Demand driven (ResourceManagers set prices)
 - Fungability (Transferring resources between domains)
- Optimal application partitioning under User criteria:
 - Minimisation of execution time
 - Minimisation of execution cost
- Open and extensible through XML
 - XML Application description
 - XML Resource description
 - XML Execution plan

High Performance Software Components

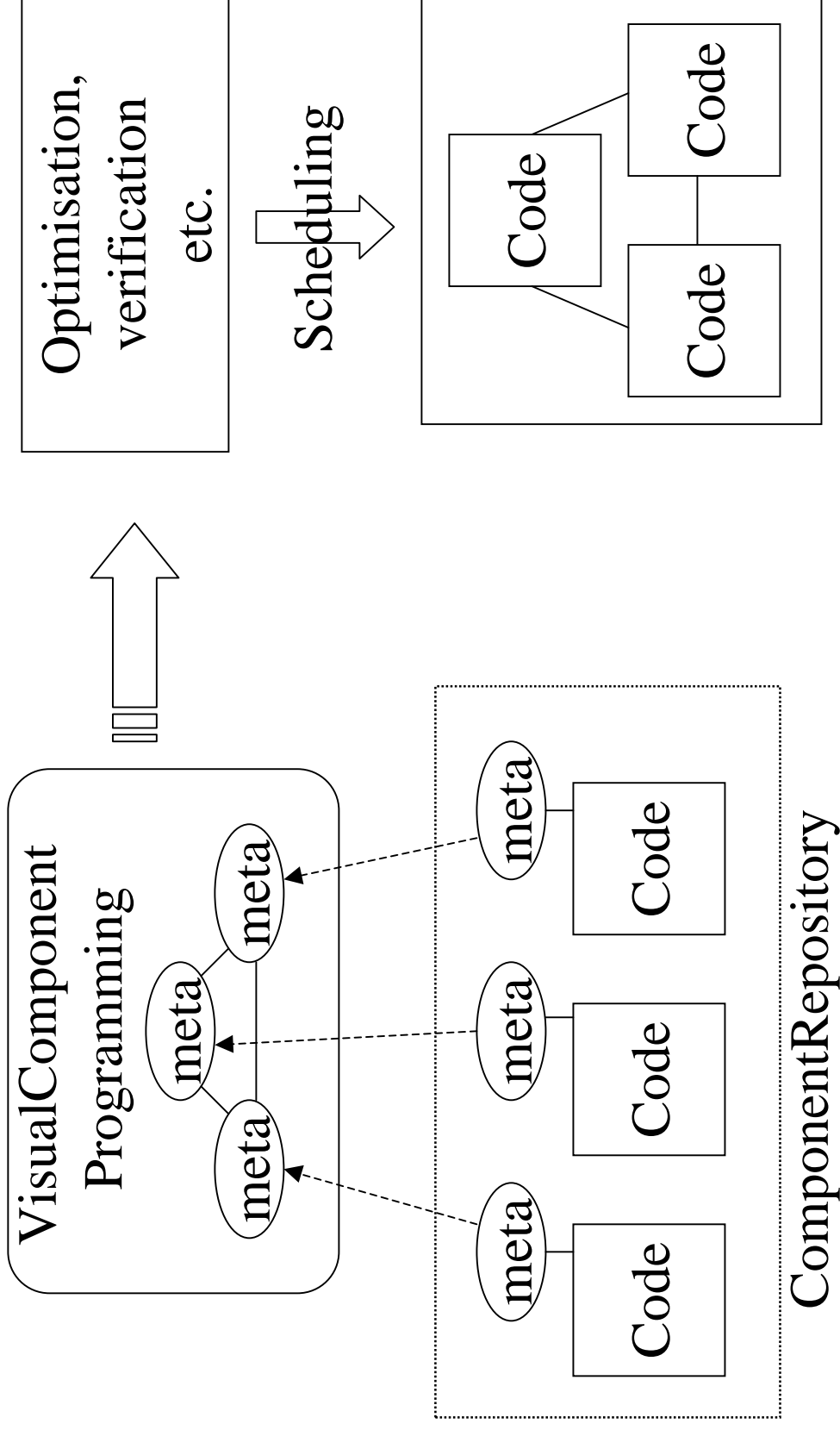
- Express an application as a composition of components (high-level abstract datatypes)
- Express the data and execution flow between the components
- Use component meta-information to inform all stages of application construction and execution
- Use CXML to describe the meta-data
- Use Java to construct the framework and interface to the assembled components

CXML as an Intermediate Language



Separating Implementation from

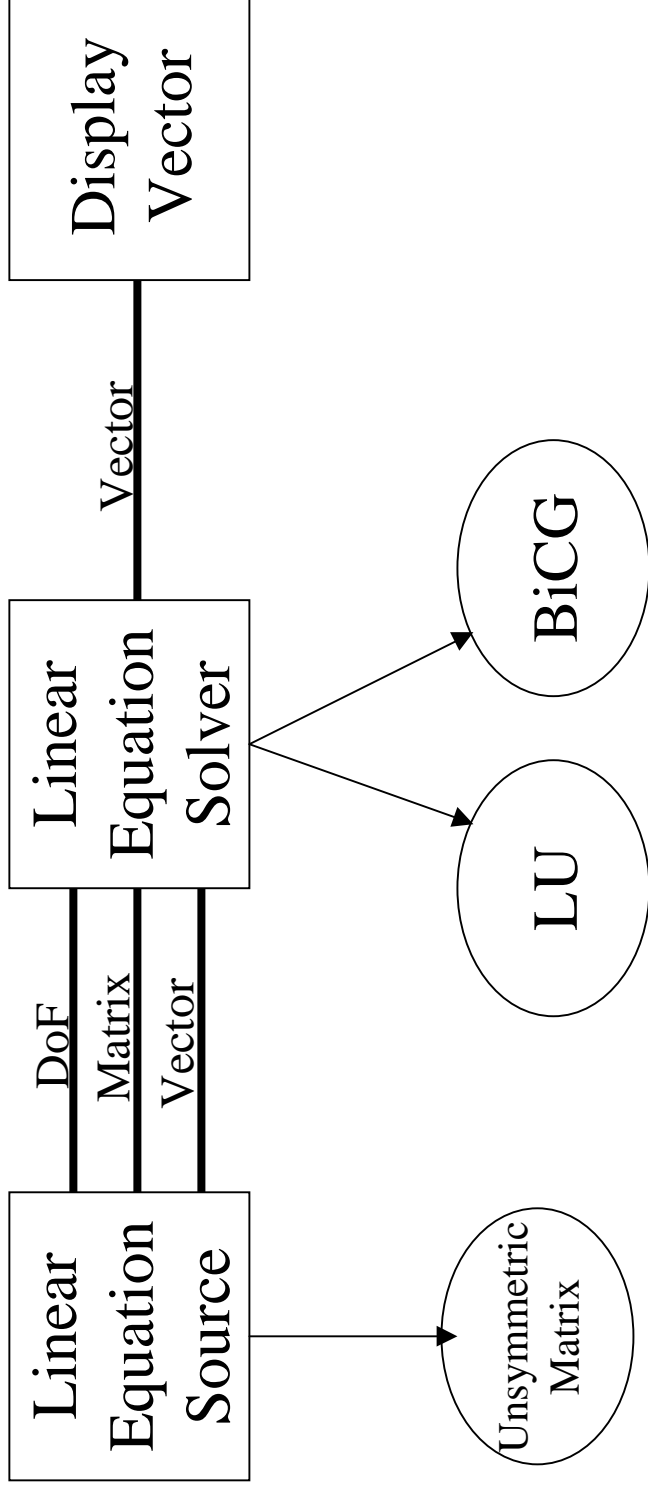
Meta-information



LinearSolverCXML - 1

```
<repository>
  <component package="icpc.denseLinearAlgebra.real"
    name="LinearEquationSourceRowsColumnsUnsymmetric" version="1">
    <propertyDefinition type="external" name="degrees of freedom" value="1000"/>
    <port behaviour="OUT" objectPackage="icpc.matrix.real"
      objectName="MatrixRowsColumnsUnsymmetric" portName="matrix"/>
    <port behaviour="OUT" objectPackage="icpc.vector.real"
      objectName="Vector" portName="vector"/>
    <implementation language="java" platform="java" url="file:.">
      <action portName="matrix">
        <binding method="getMatrix"> ... </binding>
        <classPerformanceModel type="initial" url="http:" />
      </action>
    <implementation language="C" platform="Linux" url="file:."> ... </implementation>
  </component>
  <object package="icpc.matrix.real" name="MatrixRowsColumnsUnsymmetric" version="1">
    ...
    <method name="getMatrix" type="action">
      <argument mode="out" typeName="MatrixRowsColumnsUnsymmetric"
        typePackage="icpc.matrix.real" />
    </method>
  </object>
</repository>
```

Example: LinearSolver



LinearSolverCXML - 2

```
<application>
<network>
  <instance componentName="LinearEquationSourceRowsColumnsUnsymmetric"
    componentPackage="icpc.denseLinearAlgebra.real" id="1">
    <property name="degrees of freedom" value="100"/>
  </instance>
  <instance componentName="LinearSolverRowsColumnsUnsymmetric"
    componentPackage="icpc.denseLinearAlgebra.real" id="2"/>
  <instance componentName="DisplayVector"
    componentPackage="icpc.vector.real" id="3"/>

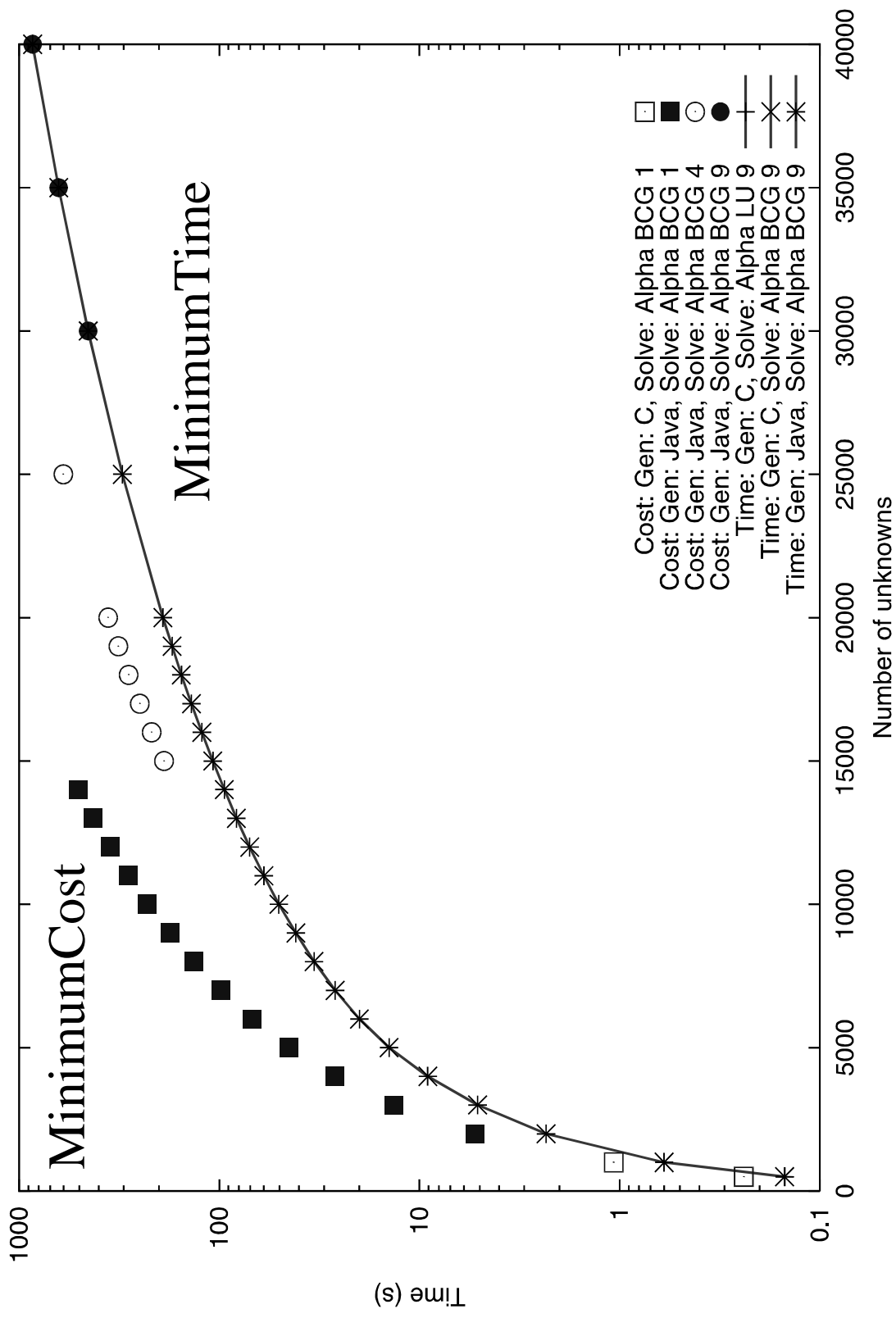
  <dataflow sinkComponent="2" sinkPort="matrix"
    sourceComponent="1" sourcePort="matrix"/>
  <dataflow sinkComponent="2" sinkPort="vector"
    sourceComponent="1" sourcePort="vector"/>
  <dataflow sinkComponent="3" sinkPort="vector"
    sourceComponent="2" sourcePort="solution"/>
</network>
</application>
```

Local Computational Community

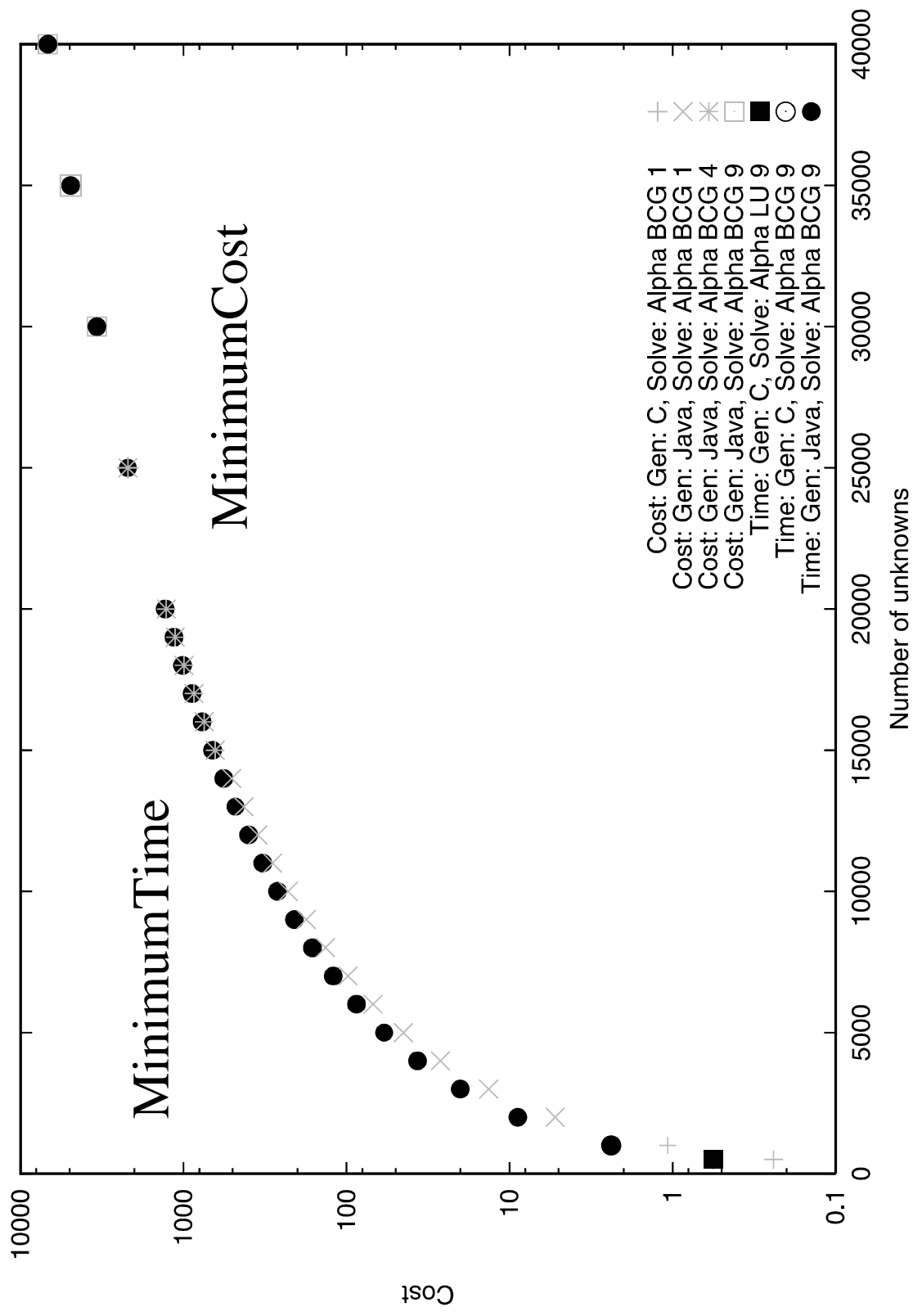
System	Processor	Processors	Language	Solution
PC	AMD	1	Java+C	LU+BCG
(Linux)	900MHz		LAPACK	LU
Atlas	Alpha 667MHz	1,4,9	ScaLAPACK	LU+BCG
AP3000	UltraSparcIII 300MHz	4,9,16	ScaLAPACK	LU+BCG

All processors have equal cost

Influence of Policy on Execution Time



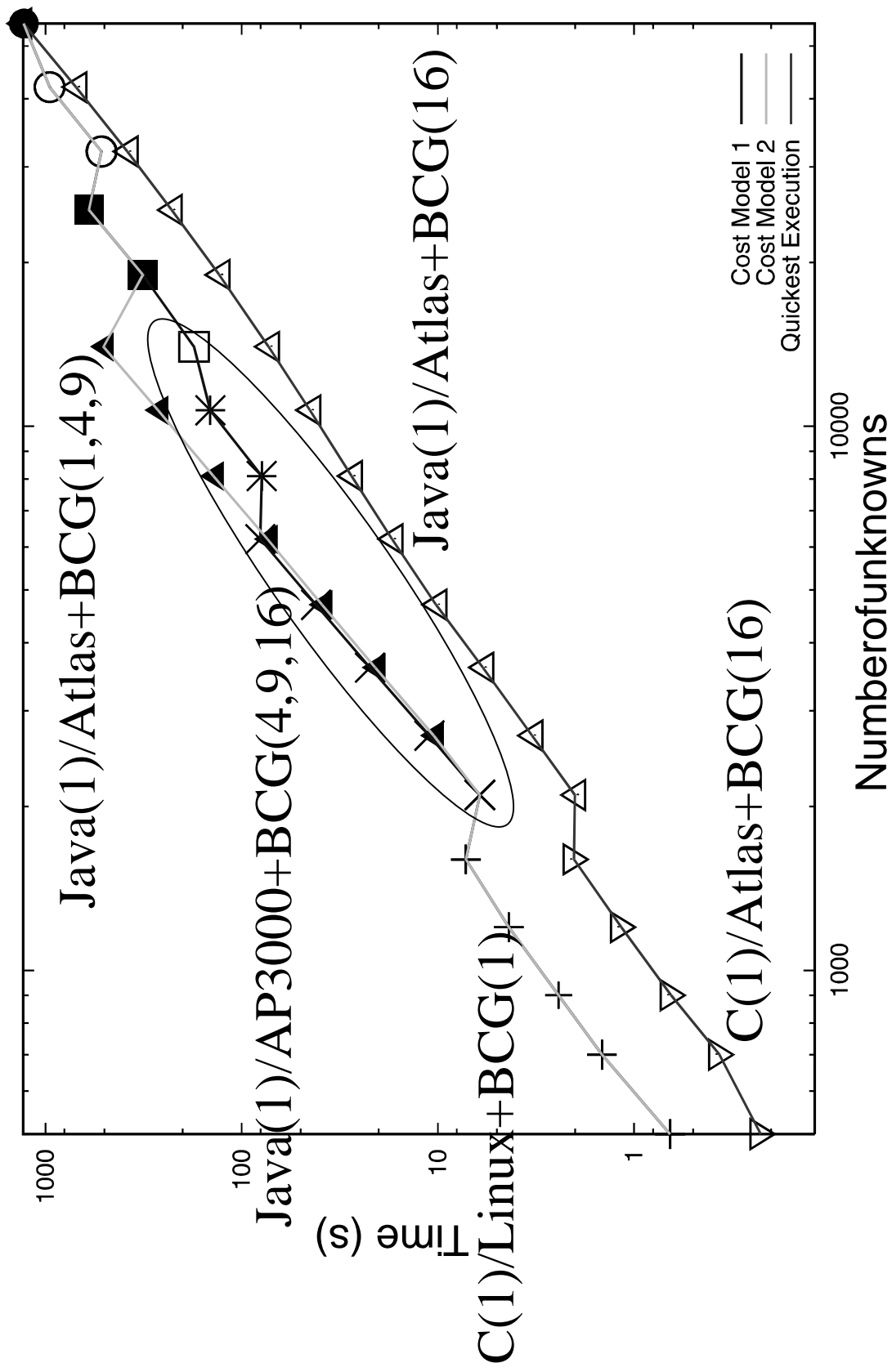
Influence of Policy on Execution Cost



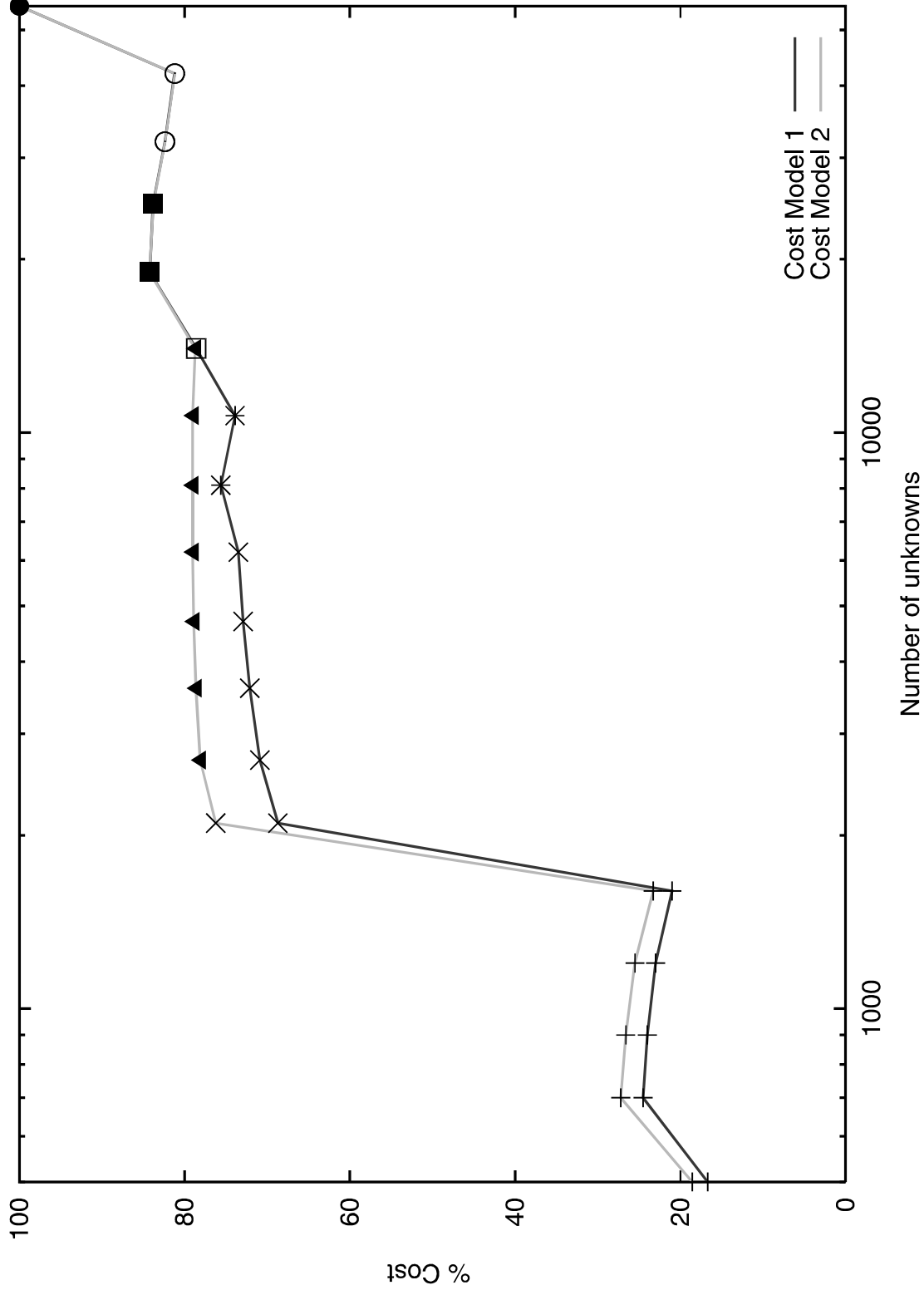
DynamicResourceCost

System	CostModel1	CostModel2
PC (Linux)	80	80
Atlas	475	425
AP3000	100	100

Influence of Cost and Policy



Cost saving relative to minimum time



Summary

- High Performance Software Components
 - Separated definition from implementation
 - Metadata relating to composition
- Computational Communities
 - Federated resources
 - Jini provides fault tolerant
- Optimise usage
 - users specify minimum time or cost
 - resource providers able to control their job mix

Acknowledgements

- ParallelSoftwareGroup:
 - JohnDarlington
 - AnthonyMayer
 - Nathalie Furmento
 - Stephen McGough
- Funding:EPSRCGR/N13371
- FurtherInformation:
 - <http://www-icpc.doc.ic.ac.uk/components>
 - email:icpc-sw@doc.ic.ac.uk