



ht

Recent Trends

- ★ *The Grid.* Use distributed hardware and software infrastructure to provide reliable, pervasive, and inexpensive access to computational resources irrespective of their physical location or access point.
- ★ *Application Service Provider.* Provide or sell computational services via web interface.
- ★ *Virtual Organization.* A group of people or institutions with some common purpose or interest that need to share (computer) resources to further their objectives.

Recent Software Technologies

- ✱ *Jini*. Java-based infrastructure for distributed service provision.
- ✱ *XML*. Standard way of storing structure data in a text file.
- ✱ *XSLT*. High level data transformation language, often used to transform XML files.
- ✱ *Semantic Web*. Will give information a well-defined meaning to better enable computers to understand the content of documents, and thereby allow people, agents, and services to work together.



PSE Definition

Gallopoulos et al. (1991):

A PSE is "a computer system that provides all the computational facilities necessary to solve a target class of problems."

Note that this definition:

- ✱ Pre-dates the Web
- ✱ Ties a particular PSE to a particular class of problems



Vision for PSEs

- ✦ PSEs herald a new era in scientific computing, both in power and how resources are accessed and used
- ✦ PSEs will become the main gateway for scientists to access terascale computing resources.
- ✦ PSEs will allow users to access these resources from any web connection.
- ✦ PSE's support for collaborative computational science will change the pervading research culture, making it more open and accountable.



Aspects of PSEs

- ✴ Distributed
- ✴ Collaborative because interesting problems are often complex and draw on many types of expertise.
- ✴ Heterogeneity pervades network environments
- ✴ Transparency is usually desirable
- ✴ Intelligence is needed to deal with various types of complexity – especially complexity arising from the collaborative and distributed use of PSEs in heterogeneous dynamic environments.



Uses of PSEs

- ✦ Multi-disciplinary simulation
- ✦ Decision support and optimization
- ✦ Virtual prototyping
- ✦ Collaborative analysis and visualization



Problem-Solving Environments

- ✦ We can view a PSE as an environment through which end-users of a Virtual Organization exploit Grid resources.
- ✦ PSEs are application specific.
- ✦ PSEs accessed through the Web/Grid are the same as “Application Portals.”

Who is Involved in PSEs?

- ★ Application end users (scientists,engineers,etc.)
 - ✱ Solve a particular problem which is domain specific
 - ✱ Undertake "what if" investigations
- ★ Developers (programmers)
 - ✱ Create components and place them in Component Repository
 - ✱ Make new algorithms and techniques available to the end users.
- ★ Software infrastructure builders
 - ✱ Create services and interfaces
 - ✱ Provide abstractions
 - ✱ Develop standards



Intelligence in PSEs

- ✱ Collaborative and distributed computing adds new types of complexity to the software environment.
- ✱ PSEs need to be intelligent to minimize impact of this complexity on users.



Views of the Grid

- ★ Virtual Organisations form “mini-Grids” for specific purposes. Membership of VO is constrained, and resources in VO are controlled and managed.
- ★ The Consumer Grid. Services and resources can be anywhere. Important issues of dynamic resource discovery, trust, and digital reputation.



PSEs as Service Providers and Brokers

- ✴ Trend is towards network-based computing paradigm.
- ✴ Nodes offer different sets of computing services with known (or advertised) interfaces.
- ✴ Software is increasingly seen as a “pay-as-you-go” service rather than a product that you buy once.
- ✴ NetSolve is an example of a service providing numerical software.



Problem Specification and Solution

- ☀ Use either

- ☀ Visual programming environment to link software components
- ☀ High-level language specification

- ☀ Recommender systems can be used to help user choose best way to solve problem and locate software.

Software Components

- ✦ Components are specified in terms of their input and output interfaces.
- ✦ User doesn't need to know about the internal details of components.
- ✦ Different components can be scheduled on different resources.
- ✦ Components can be plugged together within a visual editor.



Software Technologies for PSEs

Wherever possible use accepted software standards and component-based software engineering.

- ✱ XML is used for interface specification and defining the component model.
- ✱ Java is used for platform-independent programming.
- ✱ Transparent interaction between distributed resources is provided by technologies such as CORBA, Globus, etc.
- ✱ Agents for user support, resource monitoring and discovery.

A Few Words on XML

- ✱ XML is a method for putting structured data in a text file
- ✱ XML looks a bit like HTML but isn't HTML
- ✱ XML is text, but isn't meant to be read
- ✱ XML is a family of technologies
- ✱ XML is verbose, but that is not a problem
- ✱ XML is license-free, platform-independent and well-supported

Example XML for Component

```
<?xml version="1.0" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="example.xsl"?>
<component>
  <name>mdrun</name>
  <preface>
    <name>Molecular Dynamics</name>
    <author>
      <name>David W. Walker</name>
      <email>david@cs.cf.ac.uk</email>
    </author>
  </preface>
  <help>http://www.cs.cf.ac.uk/User/David.W.Walker/</help>
  <interface>
    <input position="2" tag="temperature" type="float" value="0"></input>
    <input position="1" tag="density" type="float" value="0"></input>
    <output position="3" tag="energy" type="float*"></output>
  </interface>
</component>
```

More XML/XSLT examples

- ✱ A simple address book.
- ✱ A more complicated example related to genealogy.

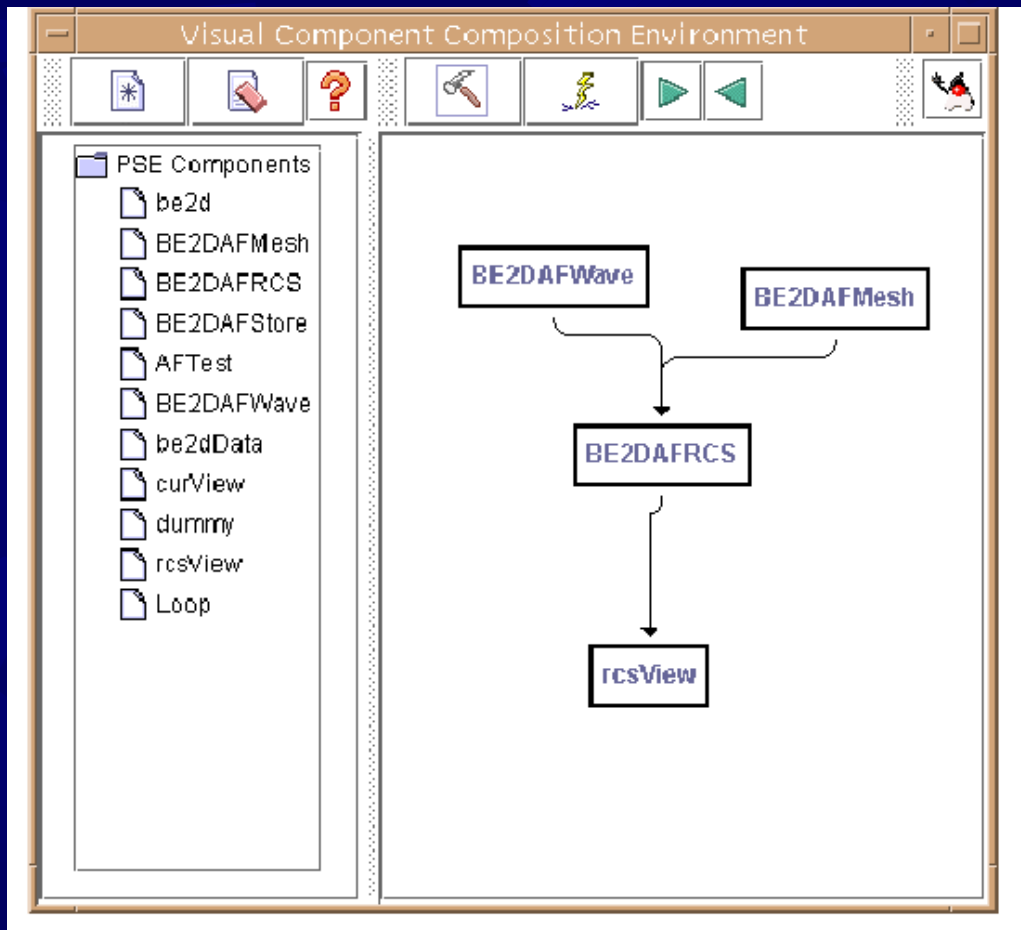


Collaborative Code Development Environment

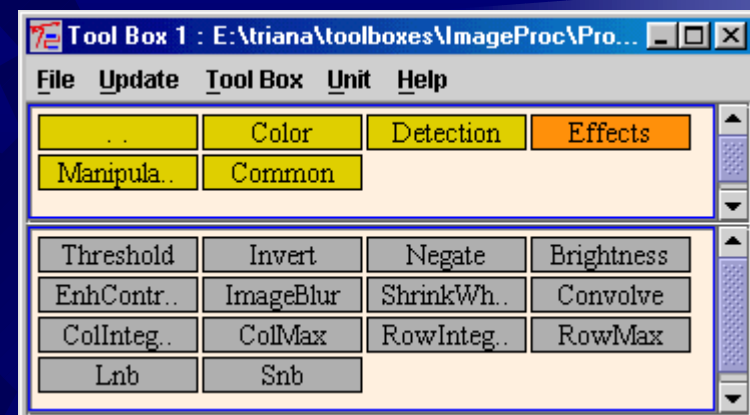
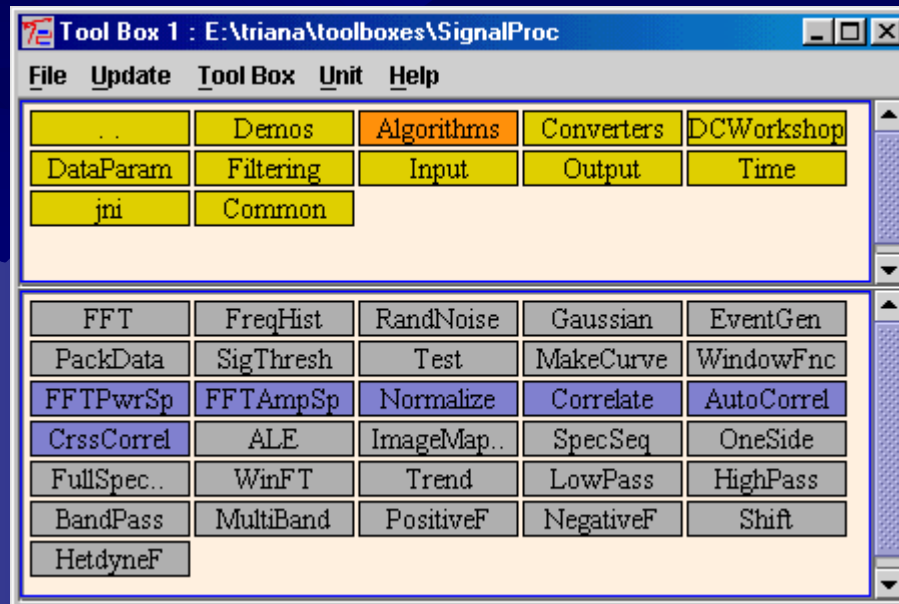
- ✦ The collaborative code development environment uses a visual programming tool for seamlessly integrating code from multiple sources.
- ✦ Applications are created by plugging together software components.
- ✦ Different developers can place their components in a shared repository.
- ✦ Legacy codes in any major scientific programming language can be handled.

Plug and Play Components

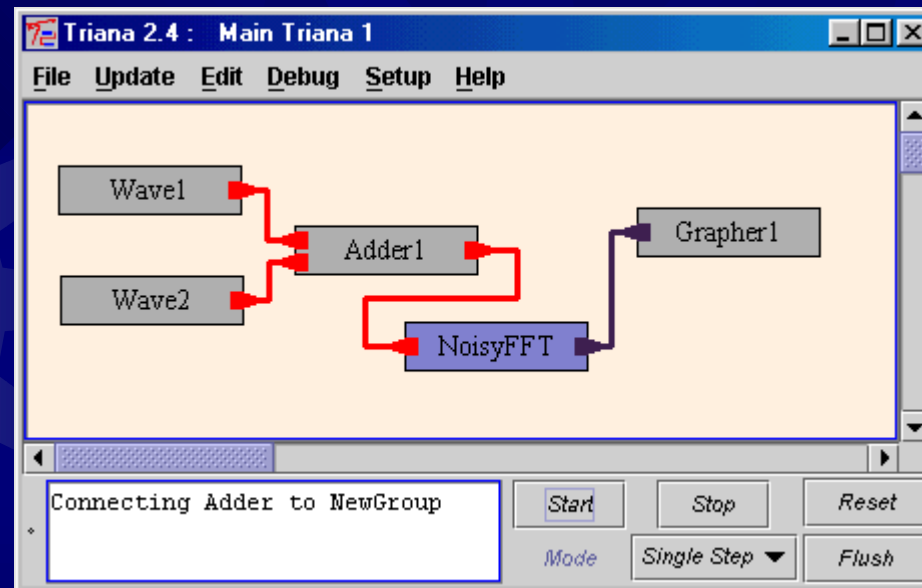
- Can link the output of one component to the input of another.
- Store components in a repository.



See Triana for example <http://triana.co.uk/>

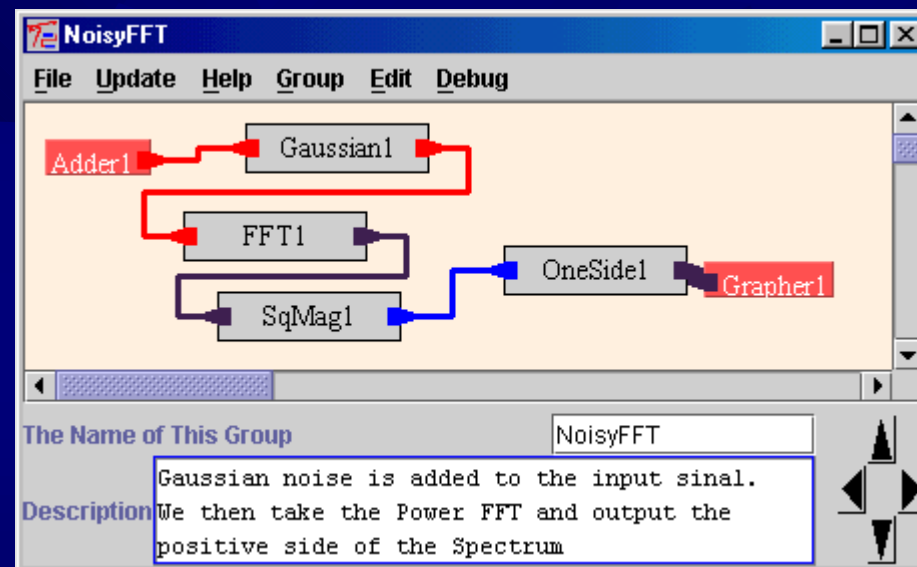


Triana's Workspace Window



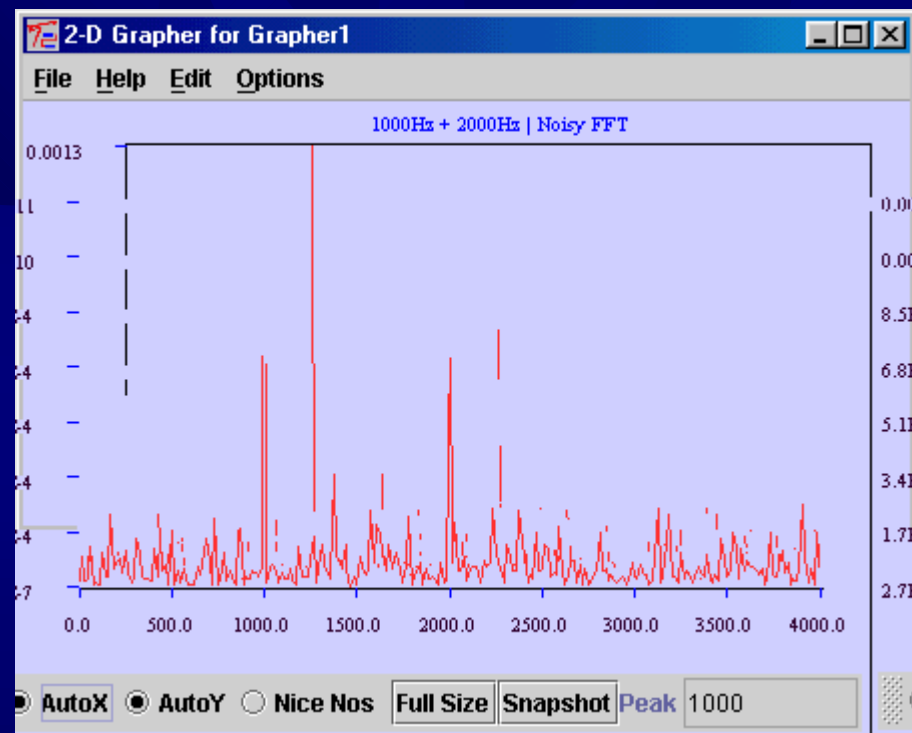
Here, two sine waves, 1kHz and 2kHz are added and then passed to a NoisyFFT component and displayed with the Grapher

Triana Groups



In Triana, any units can be grouped and groups can contain subgroups. Here we show the NoisyFFT group used in our network

Graphical Output



FOR MORE INFO...

See website : <http:// triana.co.uk/>

Scheduling

- ☀ Need to schedule components on distributed resources to achieve goals such as
 - ☀ Minimum execution time, or
 - ☀ Maximum throughput
- ☀ Achieving fair and efficient transparent access to resources is a difficult problem in a dynamic environment.



Scheduling Issues

- ☀ Need to be able to monitor hardware resources.
- ☀ Need to be able to locate software resources.
- ☀ May need to negotiate for use of resources.

Two Views of Components

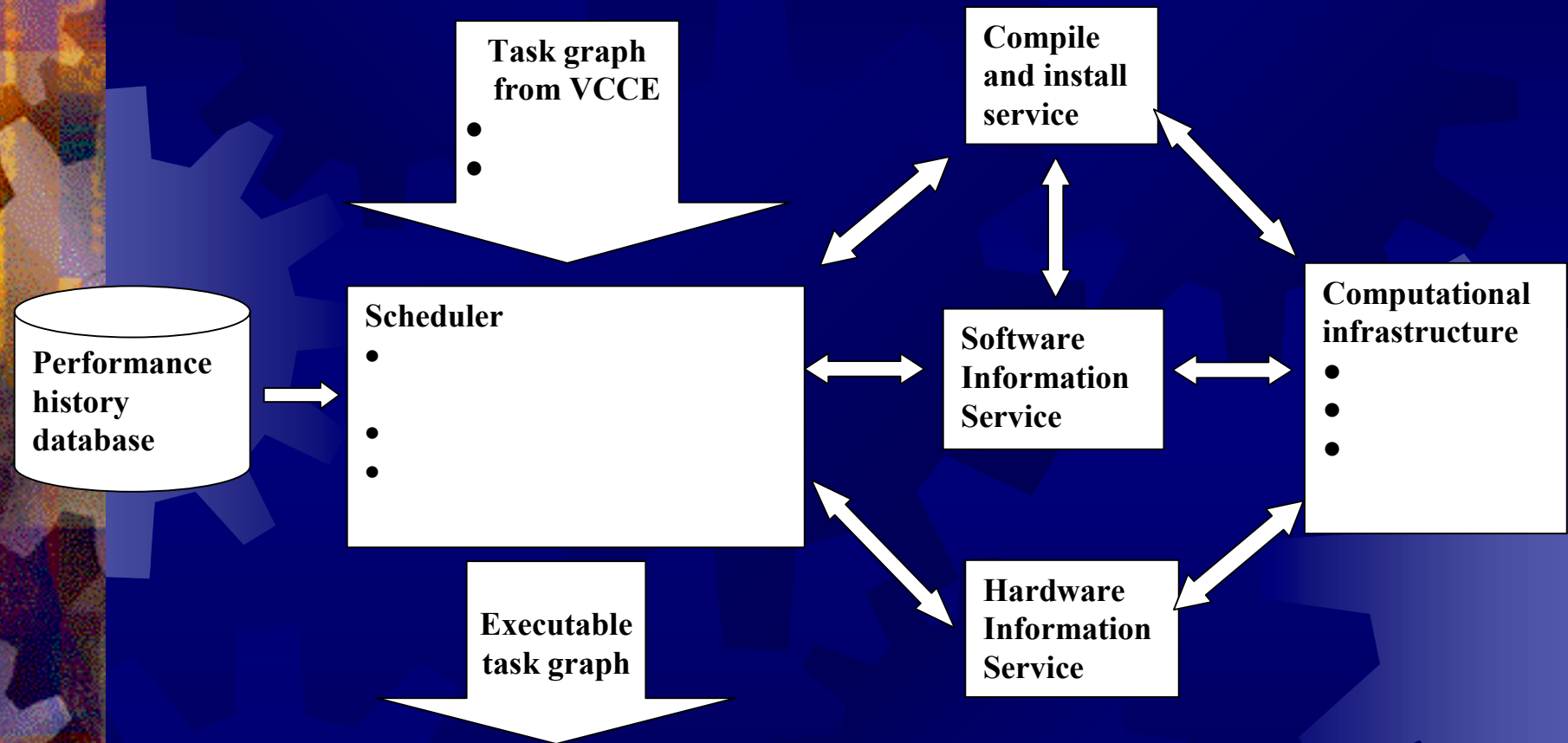
- ✦ A component is an executable that runs on a certain specified machine.
- ✦ A component can be viewed as a contract. It says: *“If you give me these inputs then I’ll give you these outputs.”*
- ✦ In the second case the component is not tied to any particular executable. Problem specification is separate from service provision.



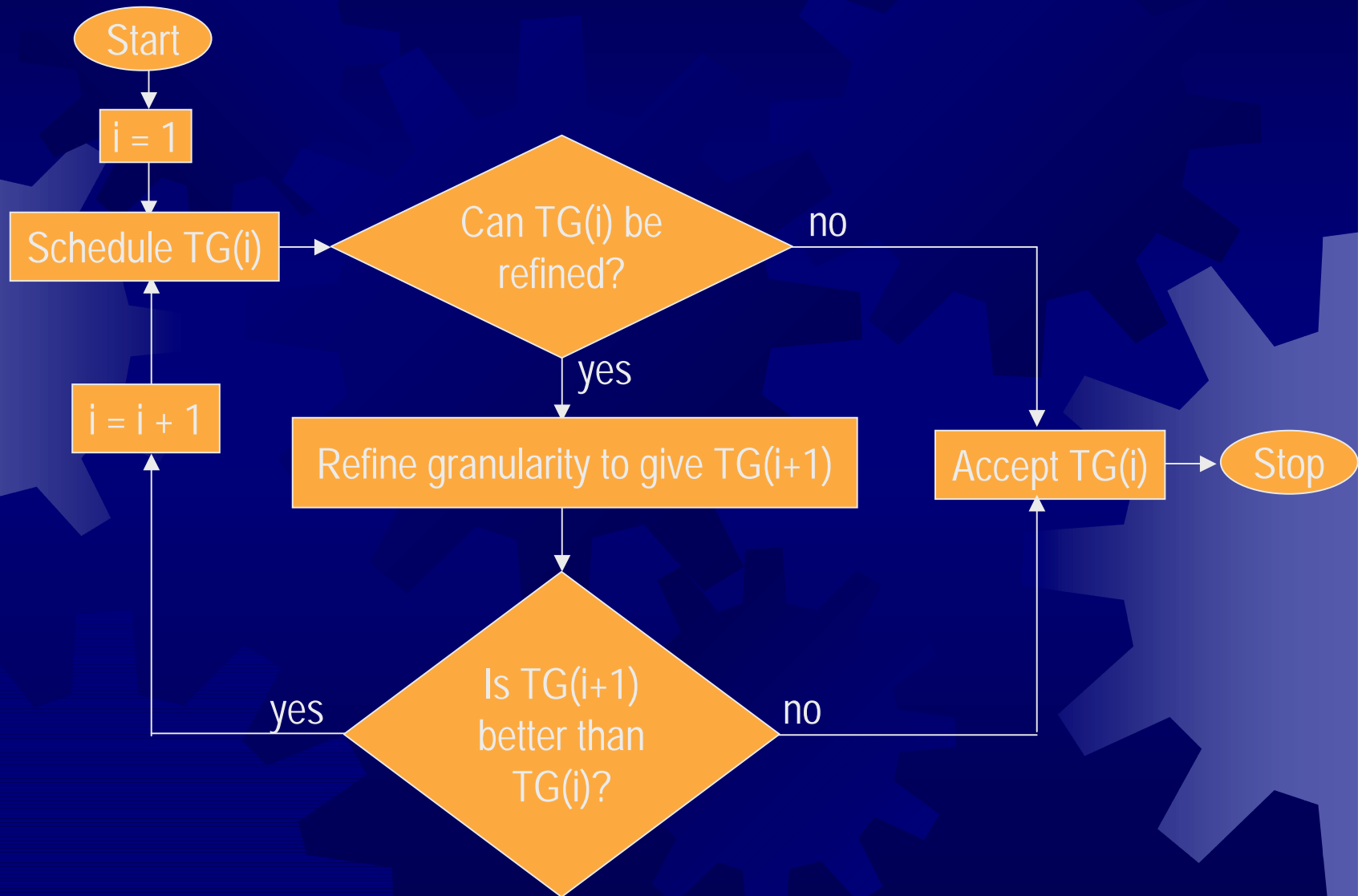
Binding Service Requests to Resources

- ✱ In a fully transparent system the scheduler would decide where components execute based on:
 - ✱ Availability and performance of resources
 - ✱ Cost and time constraints
- ✱ This is a hard problem.
- ✱ Possible solution is to supply “hints” about where it can run in a component’s XML specification.

Scheduling Services for a PSE



Selecting Granularity





Intelligent Interfaces

- ✦ Want to provide the end-user with a view of the interface that matches their degree of expertise and requirements.
- ✦ Interface should provide user support for wide range of different users.

Jean-Paul Sartre's View:

“Hell is other peoples’. Software”

- ✱ JPS was referring to the legacy code problem
- ✱ How to use existing code as components in a distributed environment



Legacy Code

- ✦ Want to automatically componentize legacy software.
- ✦ Especially large libraries of scientific and numerical software.
- ✦ Typical approach is to “wrap” each piece of code for use as a component.

JACAW

- ✴ **J**ava-**C** Automatic **W**rapper
- ✴ JACAW automatically generates the JNI interface for making C native method calls from Java.
- ✴ Uses the C header files.
- ✴ Complete source code is not needed.

Legacy Issues Addressed

- ✱ Expect better performance through use of C (compared with pure Java).
- ✱ But expect some overhead associated with the JNI interface due to copying of arguments.
- ✱ Makes use of existing validated code.
- ✱ JACAW can be applied automatically to large libraries of C software.

JACAW Example

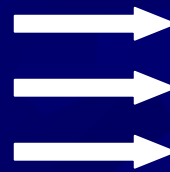
- ✴ Suppose the header file util.h contains:
signed int rand ();
void bubblesort (int, int[]);
- ✴ JACAW builds class files for the two routines and the JNI interface code for them.

What JACAW Does

util.h



Create Java files



utilJ_bubblesortM.java
utilJ_randM.java
utilJ.java

Compile Java files



utilJ_bubblesortM.class
utilJ_randM.class
utilJ.class

Create the header file



utilJ.h

Create the C file




utilJCall.c

Build the library



libutil.so



```
public class utilJ_bubblesortM{
    boolean needCopy = true;
    int arg0;
    int[] arg1;
    public utilJ_bubblesortM(int arg0, int[] arg1){
        this.arg0 = arg0;
        this.arg1 = arg1;
    }
    public void nativeCall(){
        utilJ.bubblesort(this.arg0, this.arg1, this.needCopy);
    }
    public int getArg0(){
        return this.arg0;
    }
    public int[] getArg1(){
        return this.arg1;
    }
    public void setNeedCopy(Boolean needCopy){
        this.needCopy = needCopy;
    }
}
```


utilJ.java

```
/* THIS CODE IS AUTOMATICALLY CREATED BY JACAW */
```

```
public class utilJ{  
    public native static int rand();  
    public native static void bubblesort(int arg0, int[] arg1, boolean needCopy);  
    static {  
        System.loadLibrary("util");  
    }  
}
```

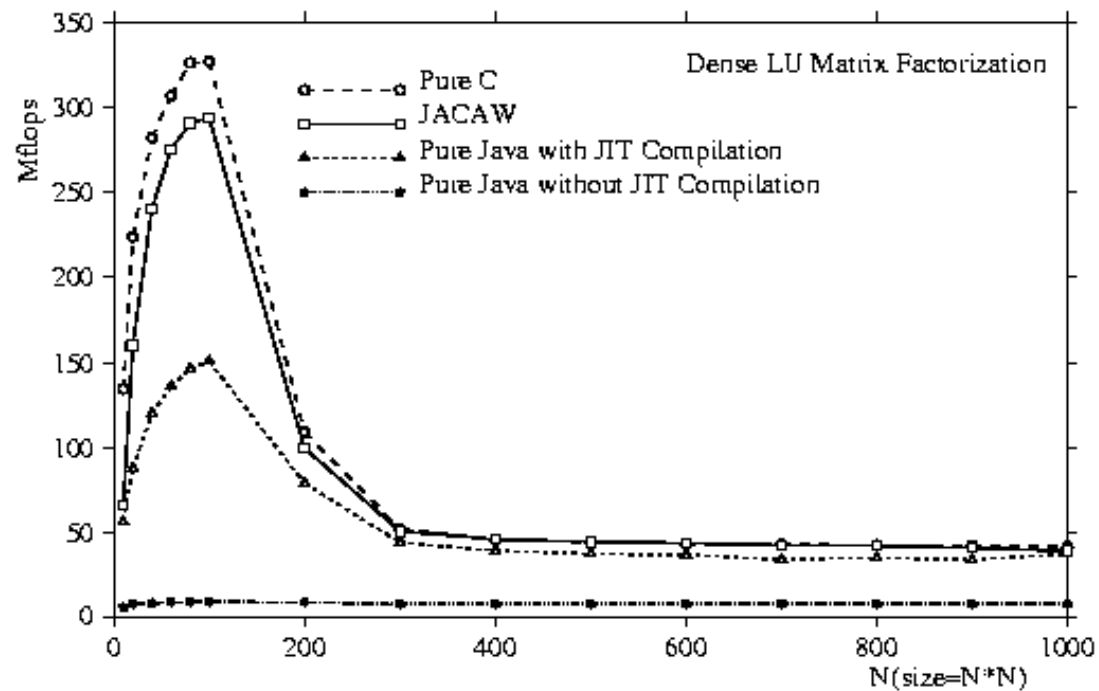
utilJcall.c

```
#include <jni.h>
#include "/home/scmyh/test/JACAW/try1/util/util.h"
JNIEXPORT void JNICALL Java_utilJ_bubblesort(JNIEnv * env,
    jclass cls, jint arg0, jintArray arg1, jboolean needCopy) {
    /* This is a native call for method: void bubblesort(int, int [])*/
    jint* arg1P;
    int mode;
    arg1P = (*env)->GetIntArrayElements(env, arg1, NULL);
    bubblesort(arg0, arg1P);
    if (needCopy == JNI_TRUE) mode = 0;
    else mode = JNI_ABORT;
    (*env)->ReleaseIntArrayElements(env, arg1, arg1P, mode);
}
```

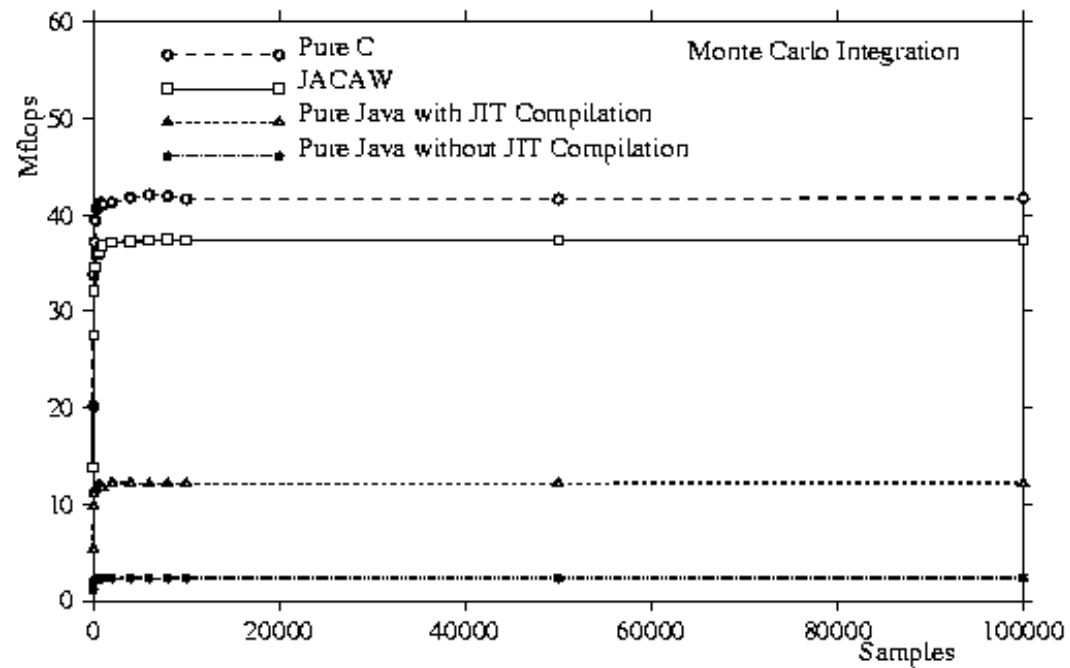
Benchmarking

- ✱ Benchmarks based on SciMark 2.0
- ✱ JACAW-generated code has good performance when ratio of number of floating-point operations to size of input arguments, N , is increasing function of N .

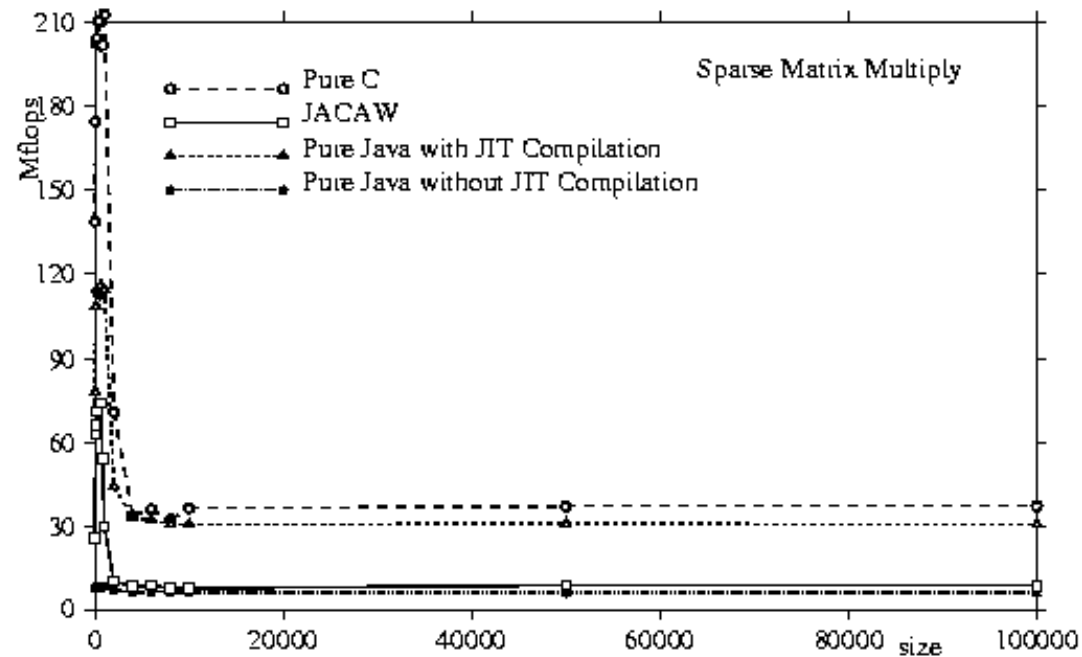
LU on dual-PIII Linux Machine



Monte Carlo Integration



Sparse Matrix-Vector Multiply






Software Agents in PSEs

Agents can be used for:

- ✦ *Software propagation.* Mobile agents can automatically install software on machines for which they are authorized.
- ✦ *Resource monitoring and discovery.* Agents can discover new resources and interact to monitor them.
- ✦ *Code migration.* Agents can carry the code to the data rather than vice versa.



The “agent” abstraction

Use of agents as core units within a PSE

- ✱ collection of service, which can adapt to changes in operating environment
- ✱ encode behaviour rules (based on offered services) which vary with time and interactions
- ✱ a communication language and vocabulary
- ✱ a long term “goal” each agent hopes to pursue



Agent services

- ✱ Agents can offer a specific service, or it can be part of infrastructure
- ✱ Specific Services: matrix solver, PDE solver
- ✱ Infrastructure Services: security manager, storage manager, checkpointing service
- ✱ Collectively, these specific and infrastructure services we call "the Agent Grid", and can organise these into layers

Agent based PSEs

- ✦ Agents offer particular services to application scientists, which change with usage
- ✦ Development environments and numerical solvers are wrapped as agents
- ✦ Interaction based on a domain vocabulary which is problem specific
- ✦ A management vocabulary that is problem independent



High Level View of Network Computing

- ✱ Services are advertised on the network
- ✱ A service typically consists of
 - ✱ A component that actually provides the service, and
 - ✱ An agent that mediates access to the service.
- ✱ Scheduler must be able to locate services and then schedule use.

Summary

- ✱ PSEs must be intelligent to deal with complexities of collaborative and distributed computing.
- ✱ Agent technologies will probably play a role in making PSEs intelligent.
- ✱ What role will the Semantic Web play in PSEs and The Grid in general?



The Semantic Web and Agents

- ✱ Agents will be able to cooperate in the handling of data with semantics.
- ✱ Agent-mediated service discovery will use the Semantic Web to understand the service offered and how to use it.
- ✱ This type of service discovery will be an important part of next generation PSEs, and hence of the Grid.

Final Remarks

- ✱ PSEs provide an integrated computational environment for solving problems in a particular domain.
- ✱ PSEs provide problem solving power without the end user needing to be an expert in the technologies used to effect the solution.
- ✱ PSEs will be implemented on top of Grid services
- ✱ How will Grid services be implemented?

More Final Remarks

- ✦ Collaboration, intelligence, and visualization are important aspects of PSEs.
- ✦ Components are a good way of encapsulating software services.
- ✦ XML is an important part of the software infrastructure of PSEs.
- ✦ Resource/service discovery and scheduling in Grid environments are critical areas for further research.
- ✦ Semantic Web may provide basis for future PSEs.