The University of Reading

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY

A Parallel Plug-in Programming Paradigm

Ronald Baumann[1,2], Christian Engelmann[1,2], and Al Geist[2]

[1] Department of Computer Science
The University of Reading, Reading, UK

[2] Computer Science and Mathematics Division,
Oak Ridge National Laboratory, Oak Ridge, USA

Office of Science
U.S. DEPARTMENT OF ENERGY

Presentation Overview

- Motivation/Background
  - Pluggable component architectures
  - Harness Distributed Virtual Machine (DVM)
  - Common Component Architecture (CCA)
- Proposed approach
  - Parallel plug-in abstraction
  - Parallel plug-in types, coordination, and fault tolerance
- Prototype Implementation
  - Developed parallel plug-in suite
  - Two computational applications using parallel plug-ins

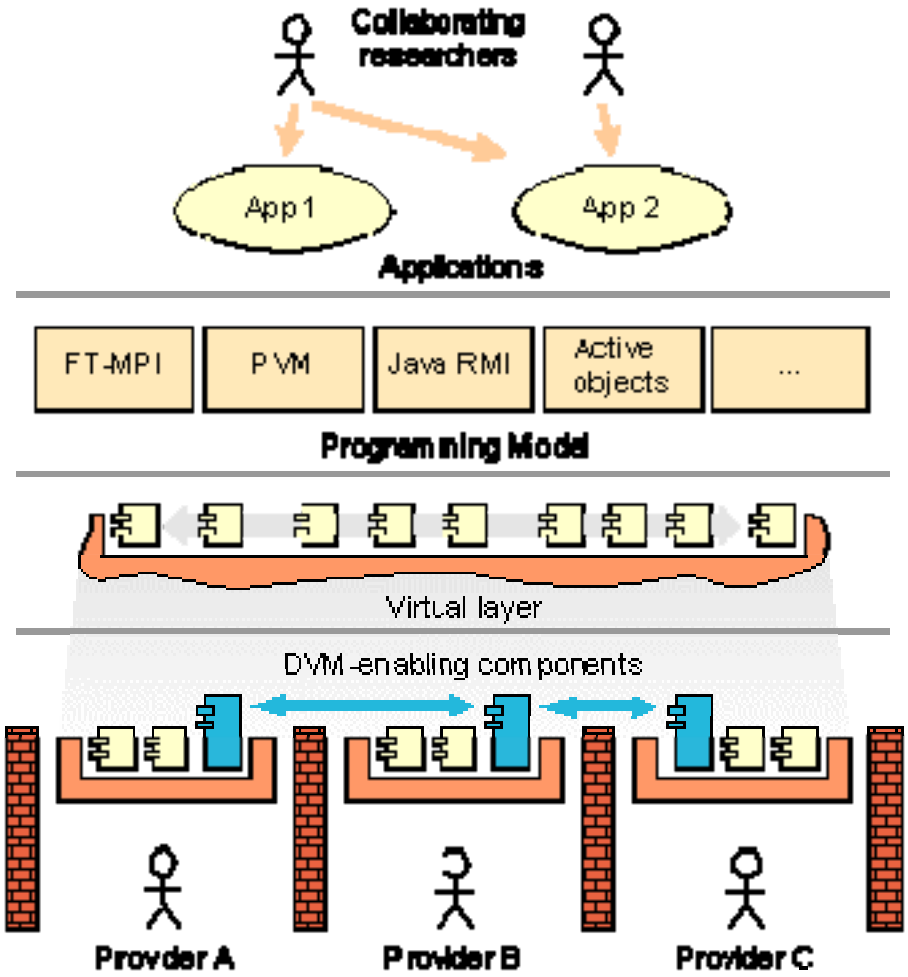
Pluggable Component Architectures

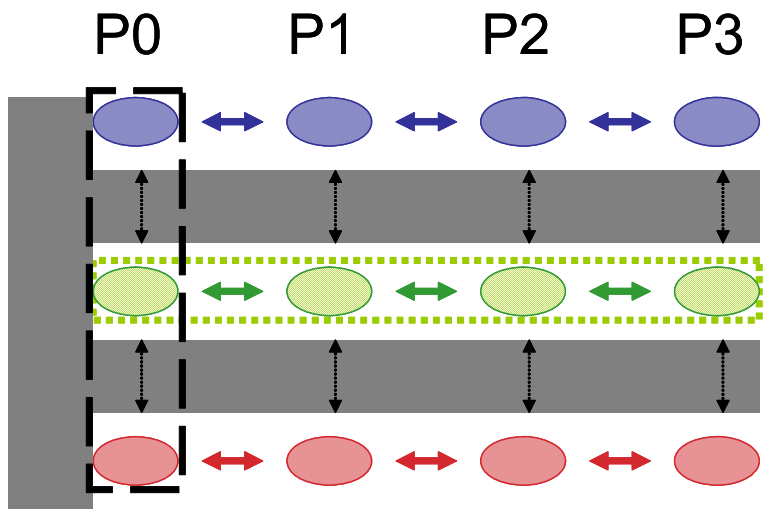
- Allow assembly of applications from individual software modules based on clearly defined programming interfaces.
- Component architectures for scientific computing
  - Assembly of SPMD/MPMD applications from components
  - Examples:
    - Harness Distributed Virtual Machine (DVM) - in research
    - Common Component Architecture (CCA) - in production
    - Others exist with similar properties, some are for middleware only (LAM/MPI, Open RTE)

DVM vs. CCA

- Harness Distributed Virtual Machine (DVM)
  - All processes cooperate in a virtual machine environment
  - Easy to implement cooperating plug-ins via DVM
  - Fault tolerance via plug-in checkpoint/restart by DVM
  - Scalability limit due to DVM abstraction
- Common Component Architecture (CCA)
  - Each process individually manages its own components
  - Limitations for cooperating plug-ins using MPI/PVM
  - Fault tolerance via process-level checkpoint/restart
  - Highly scalable due to localized configuration

Harness DVM vs. CCA





Components: Blue, Green, Red

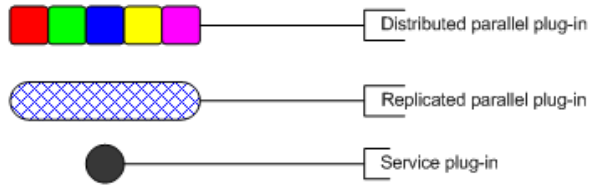
Framework: Gray

Parallel Plug-In Abstraction

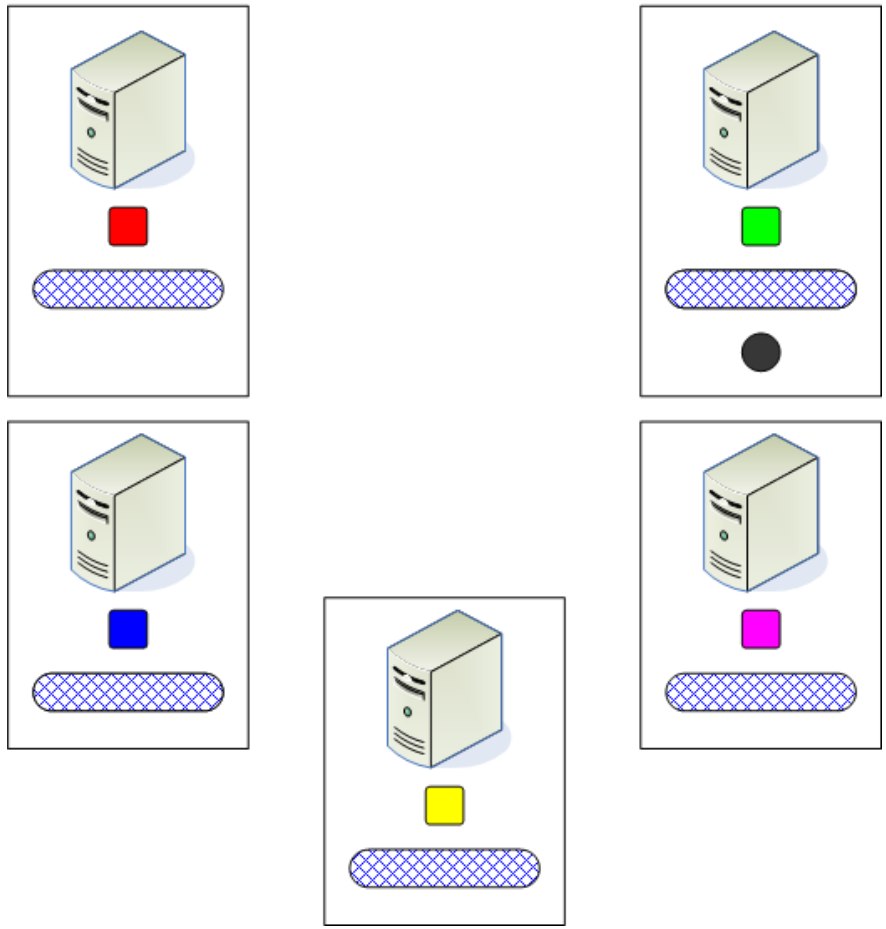
Similar to a parallel program, a parallel plug-in

- Consists of a set of cooperating concurrent plug-ins
- Communicates via message passing and/or RMI
- Allows the assembly of parallel applications
- Uses parallel programming models (SPMD/MPMD)
- Is supported by an environment (component framework) that is aware of its parallel nature
- Provides a building block (component) with a clearly defined interface

Parallel Plug-In Types



- Singleton (or service) plug-in
- SPMD (or replicated) plug-in
- MPMD (or distributed) plug-in
- Communication via PVM/MPI
  - Plug-in naming service needed for message routing
- Communication via RMI
  - Participating plug-ins are exported as objects



Parallel Plug-In Coordination

- Loading (initialization), unloading (finalization), task and data distribution, and fault tolerance
- May be performed by a service plug-in or by the pluggable component framework itself
- Self-configuring parallel plug-ins may perform task and data distribution, and fault tolerance
- Service plug-ins may provide *different templates* for parallel plug-in coordination (e.g. SPMD/MPMD)

Parallel Plug-In Fault Tolerance

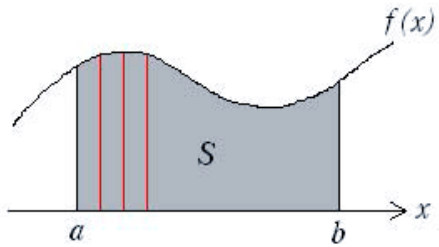
- 1. detection, 2. notification, and 3. reconfiguration
- Detection and notification are typically performed by the communication sub-system
- Reconfiguration takes place within the parallel plug-in programming scope, using:
  - Checkpoint/restart of plug-in state
  - Self-healing algorithms
- Service plug-ins may provide *different templates* for parallel plug-in fault tolerance

Prototype Implementation (1/2)

- University of Reading Master's thesis project
- Harness runtime environment (RTE) w/o DVM:
  - Lightweight backbone to (un)load individual plug-ins
  - RTE = multi-threaded process residing on a single node
  - Multiple Harness RTEs and individual remote plug-ins communicate via RMI using the RMIX framework
- Parallel Plug-In Manager (PPM)
  - Harness service plug-in
  - Provides two templates for coordinating parallel plug-ins (SPMD/MPMD)

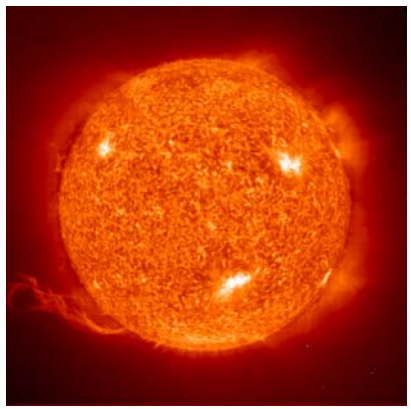
Prototype Implementation (2/2)

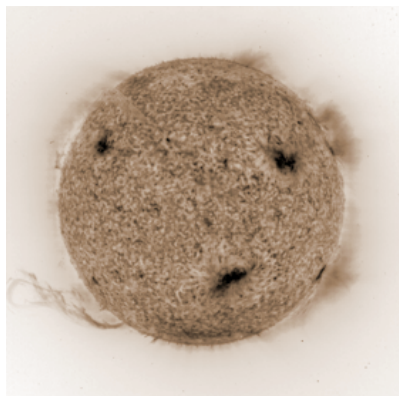
- Two different parallel plug-ins:
  - Monte Carlo integration based on SPMD (bag-of-tasks)
  - Image processing based on MPMD (pipeline)



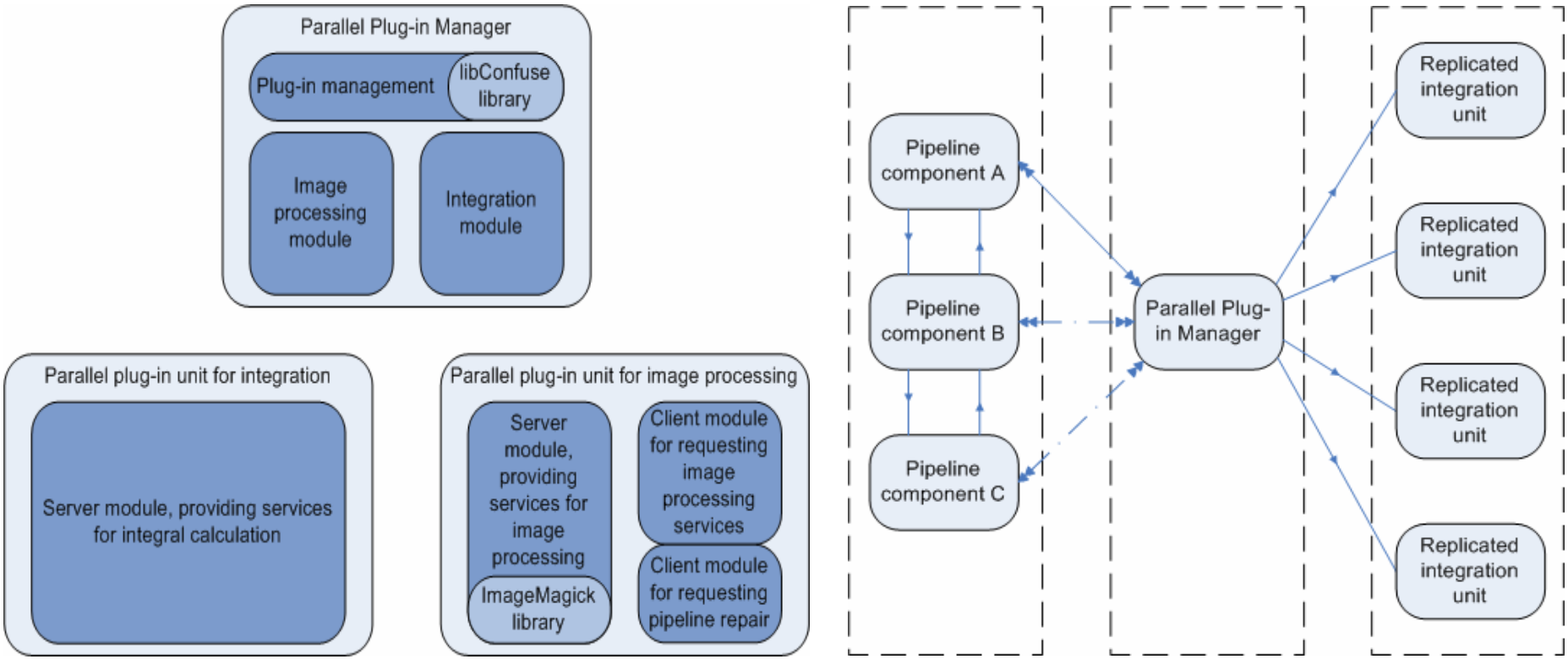
$$I = \int_a^b g(x)\mathrm{d}x$$

$$I \approx I_{MC} = \frac{b-a}{n}\sum_{i=1}^{n} g(x_i)$$

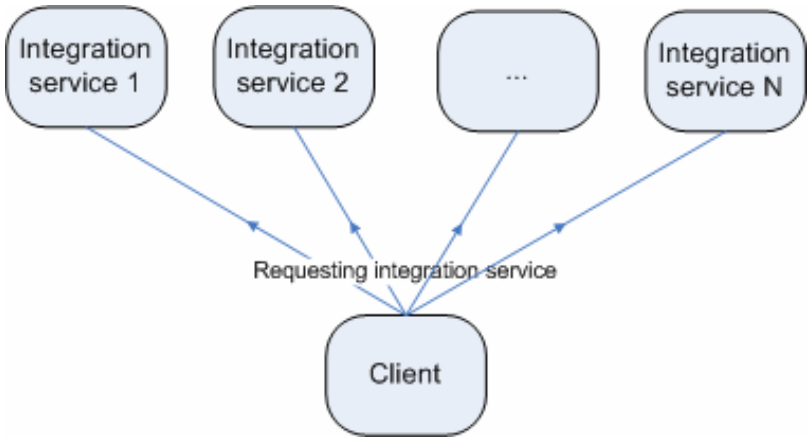




Developed Parallel Plug-In Suite



Monte Carlo Integration based on SPMD



- PPM loads the parallel plug-in on all available nodes
- PPM accepts partial loading success
- PPM performs task/data distribution
- Degree of fault tolerance is *n − 1*
- *n − 1* out of *n* Harness RTEs may fail

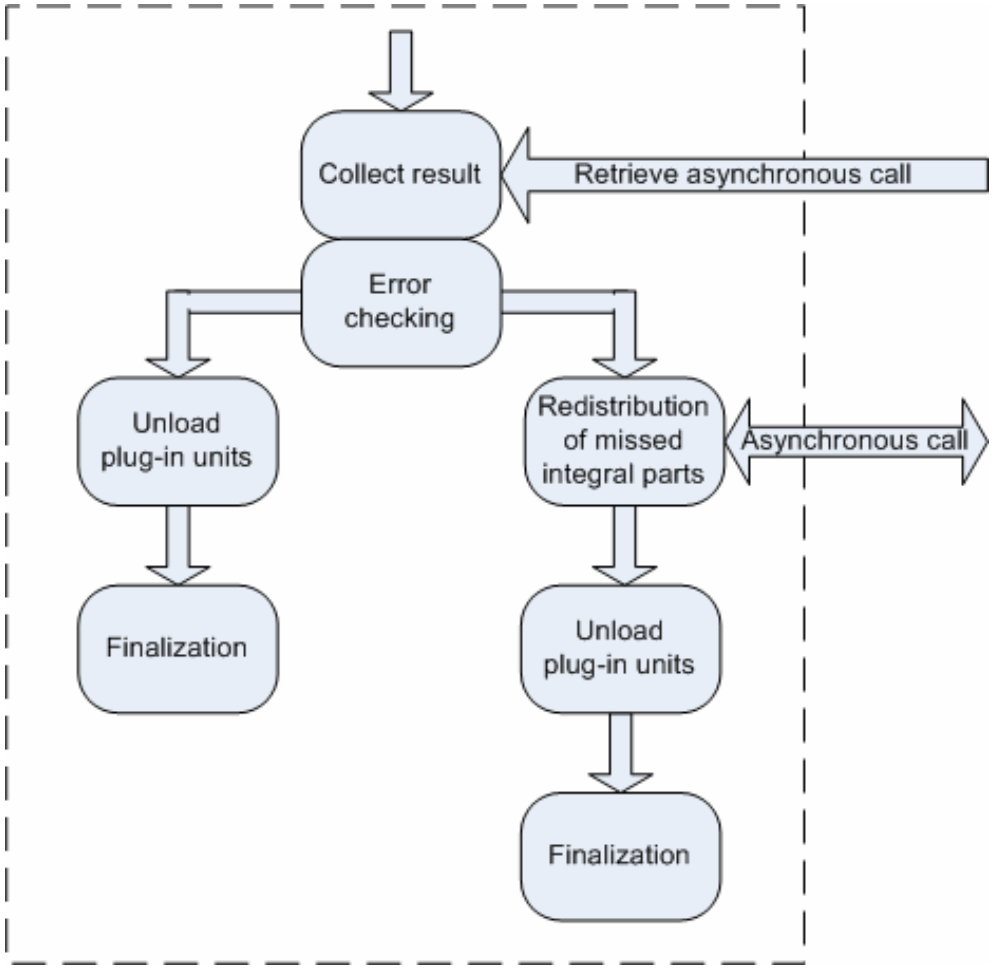
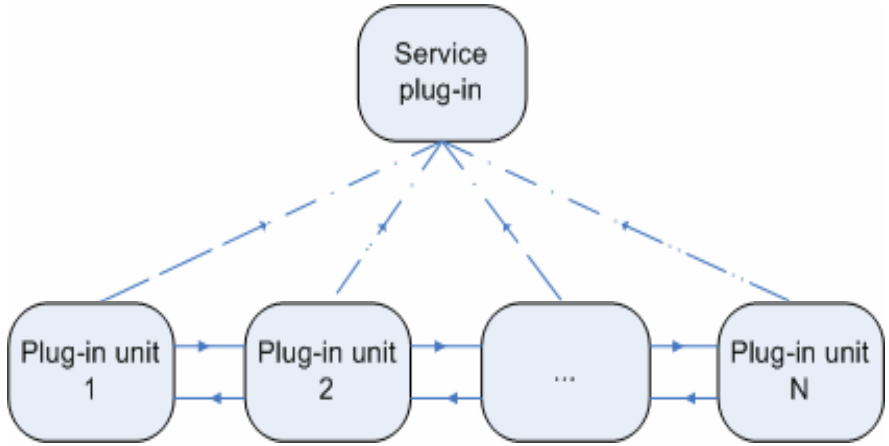
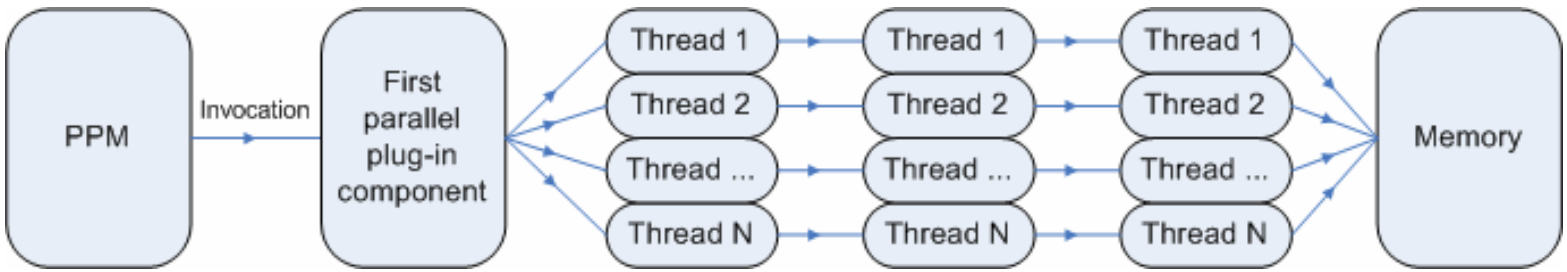


Image Processing based on MPMD

- PPM loads the parallel plug-in on the necessary number of nodes
- PPM does not accept partial loading success
- PPM performs task distribution
- Plug-in performs data distribution
- Degree of fault tolerance is *1*
- *1* out of n Harness RTEs may fail
- Multiple parallel pipelines possible





Conclusions (1/2)

- We defined the parallel plug-in abstraction, associated programming models, and programming requirements
- We demonstrated similarities and differences between parallel plug-ins and programs with regards to their programming models and execution environments
- We described a Harness-based proof-of-concept prototype for two distinct scientific application scenarios
- Further implementation details have been published in a Master's thesis

Conclusions (2/2)

- Our research indicates that the parallel plug-in programming paradigm is an appropriate design template for software component architectures in parallel and distributed scientific high-end computing
- It is our hope that this work will be eventually incorporated in some form into production-type component architectures, such as the CCA

The University of Reading

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY

A Parallel Plug-in Programming Paradigm - Questions or Comments?

Ronald Baumann[1,2], Christian Engelmann[1,2], and Al Geist[2]

[1] Department of Computer Science
The University of Reading, Reading, UK

[2] Computer Science and Mathematics Division,
Oak Ridge National Laboratory, Oak Ridge, USA

Office of Science
U.S. DEPARTMENT OF ENERGY