A Lightweight Kernel for the Harness Metacomputing Framework

Christian Engelmann and Al Geist
Oak Ridge National Laboratory

14th Heterogeneous Computing Computing Workshop 2005
What is Harness

- A pluggable, reconfigurable, adaptive framework for heterogeneous distributed computing.
- Allows aggregation of resources into high-capacity distributed virtual machines.
- Provides runtime customization of computing environment to suit applications needs.
- Enables dynamic assembly of scientific applications from (third party) plug-ins.
- Offers highly available distributed virtual machines through distributed control.
- Various experiments and prototypes (C/Java).
Harness Architecture

- Light-weight kernels share their resources.
- Plug-ins offer services.
- Support for diverse programming models.
- Distributed Virtual Machine (DVM) layer.
- Highly available DVM using distributed control.
- Highly available plug-in services via DVM.
Harness DVM Architecture

DVM Maintains Global State Via Distributed Control

Split and Merge With Other DVMs

Dynamically Customize And Extend Via Plug-Ins

H2O Kernel
- DVM
- PVM Plug-In
- FT-MPI Plug-In

HARNESS Daemon

DVM Control

User Features

Host A
Host D
Host B
Host C
Host Z
Another VM
Original Harness Kernel Design

Inside kernel (HCtl) with ring-based peer-to-peer distributed control.

Databases inside kernel for local and global info.

Peer-to-peer messaging plug-in (HCom).

Basic plug-in & external process management.

Forced/Hidden DVM programming model.

Metacomputing Framework
Improved Lightweight Kernel

- **Optional Distributed Control plug-in (DVM).**
- Only local information stored inside kernel.
- Enhanced process and plug-in management.
- Thread management.
- RMI/RPC messaging through RMIX plug-in.

-user is able to choose programming model based on actual needs: DVM, PVM, FT-MPI, client-server, etc.
Optional Distributed Control Plug-in

- Not all plug-ins need to be part of the DVM.
- User chooses if high availability is needed.
- Avoids unnecessary DVM use and associated performance impact.
- Allows loosely coupled peer-to-peer paradigms.
- Improves adaptability, versatility and usability.

• Plug-in access via DVM
• Direct plug-in access
Improved Process Manager

- Capable of controlling child processes via a separate kernel child process (forker) spawned at startup.
- Allows creation and destruction of child processes.
- Relays input to stdin of child processes.
- Optionally captures and buffers child process stdout.
- Supports sending of signals to child processes.
- Harness kernel threads may wait for child process exit.
- Typically used for remote kernel startup using ssh and for external application runs.
Enhanced Plug-in Loader

- Loads and unloads shared libraries (dlopen/dlclose).
- Initializes after loading. Finalizes before unloading.
- Allows multiple loading using unique handles.
- Offers recursive dependent plug-in (un)loading.
- Provides global symbol export or lookup (dlsym) of table with global data and function pointers.
- Supports optional plug-in versioning scheme: <version>.<age>.<revision>
- Capable of managing different plug-in versions loaded into the same kernel (without global export).
Added Thread Pool

- Allows to change minimum/maximum thread count.
- Supports variable timeouts for idle threads.
- Offers configurable kernel shutdown thread timeout.
- Capable of adjusting the maximum job queue length.
- All maximums, minimums and timeouts are reconfigurable before kernel startup and at runtime.
- Simplifies task execution in the multi-threaded kernel.
- Increasing the maximum thread count is typically used for persistent threads, like servers.
RMIX Framework

- Originally developed in Java at Emory University.
- Dynamic, heterogeneous, RMI/RPC framework.
- Pluggable providers: Sun RPC, Java RMI and SOAP.
- Support for asynchronous and one-way invocations.
- Stand-alone C variant and Harness plug-in currently in development at ORNL.
RMIX Harness Plug-in

- Reuse of Harness plug-in and thread management.
- RMIX Harness plug-in wraps RMIX base library.
- Harness plug-ins provide client and server stubs.
- Kernel stub plug-in.
- Harness plug-ins are able to communicate via RMIX.
- Further improves flexibility and heterogeneity.
Conclusions

- Improved adaptability, versatility and usability by changing to a lightweight Harness kernel design.
- Moved previously integrated distributed control service (DVM) into an optional Harness DVM plug-in.
- DVM is only used when high availability is needed.
- Improved performance by bypassing the DVM.
- Enhanced process manager to provide remote kernel startup using ssh and external application runs.
- Introduced an optional plug-in versioning scheme.
- Added thread pool to simplify task execution in the multi-threaded kernel environment.
Future Work

- Finishing the development of RMIX stand-alone C variant and RMIX Harness plug-in to further improve flexibility and heterogeneity.
- Service-level high availability features for applications, as well as for typical operating system components, such as schedulers.
- Virtualization of different underlying platforms to present a uniform programming and deployment interface.
A Lightweight Kernel for the Harness Metacomputing Framework

Questions or comments?

Christian Engelmann and Al Geist
Oak Ridge National Laboratory

14th Heterogeneous Computing Computing Workshop 2005