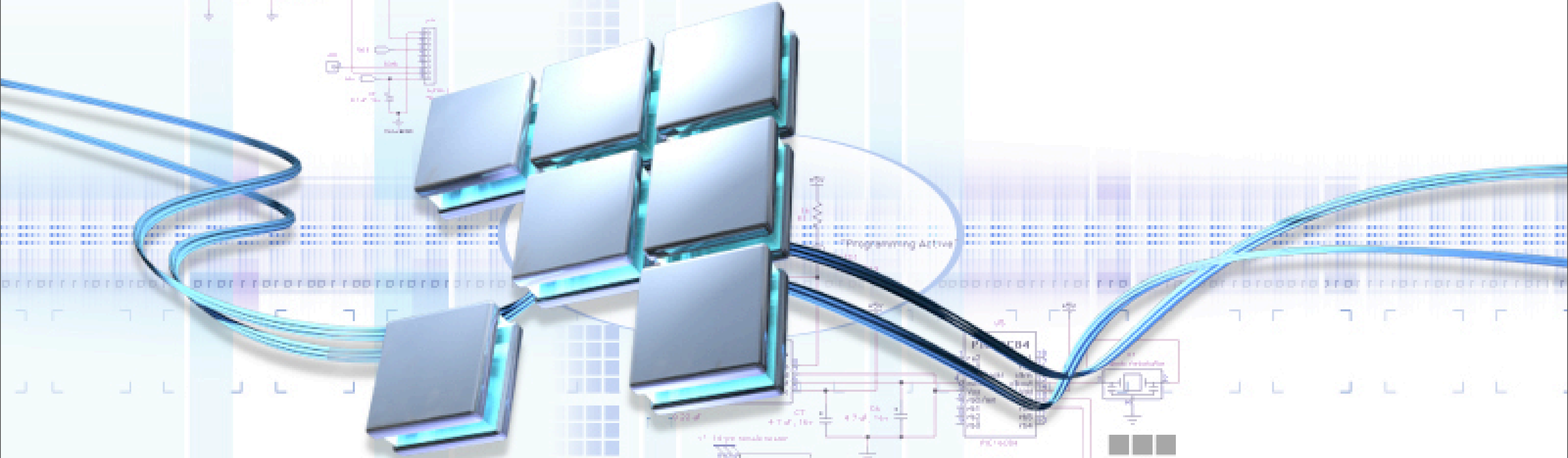


Developing a Computational Science IDE for HPC

Dave Hudak, Neil Ludban
Vijay Gadepally, Ashok Krishnamurthy

Overview

- Overview of CSIDEs: components, examples
- Software engineering benefits of CSIDEs
- HPC CSIDEs: required functionality and examples
- ParaM: experience developing HPC CSIDE
- Software engineering challenges of CSIDE users

Computational Science IDE (CSIDE)

- A suite of software tools, including
  - Numeric interpreter with high-level matrix operations
  - Domain-specific extensions (signal and image processing, control systems, operations research)
  - Graphics and visualization
  - Common user interface (usually including editor)
- Examples
  - Commercial: MATLAB, Maple, Mathematica
  - Open source
    - NumPy + SciPy + iPython + Matplotlib
    - GNU Octave + OctaveForge + GNUPlot + Emacs

Software Engineering Benefits of CSIDEs

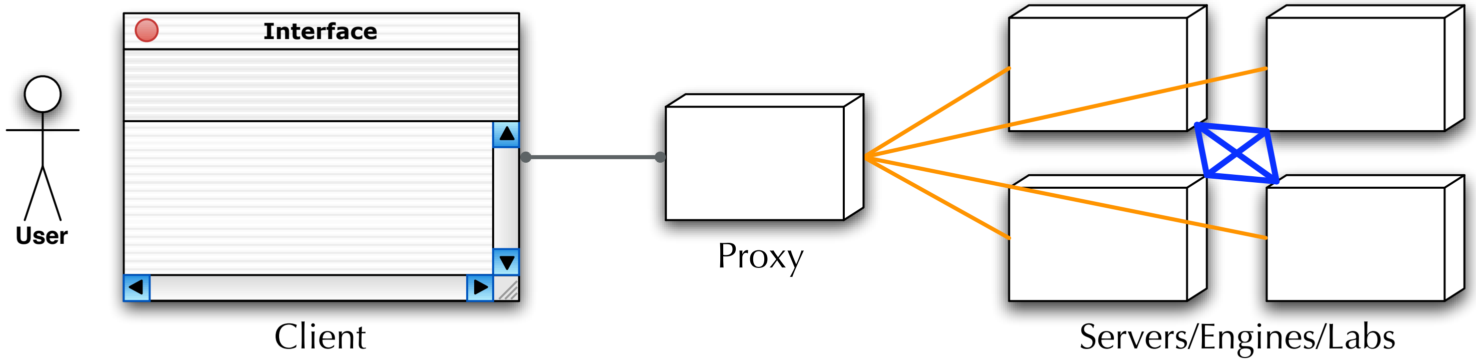
- No steep learning curve
- Fast time to first solution, lower turnaround time
- Reasons for these benefits?
  - Interactive nature of interpreter
    - Encourages incremental development
    - Immediate debugging
  - Automatic memory management
  - Domain-specific language constructs
  - Tight integration between tools

HPC Computational Science IDE (CSIDE)

- Extending CSIDE to HPC system requires
  - Job control mechanism for launching a copy of interpreter on each processor
  - Communication libraries for interpreters to exchange results
  - Mechanism for remote control and inspection
- Examples: Matlab Distributed Computing Environment (MDCE), Star-P, iPython
- HPC CSIDEs are examples of remote interactive services
  - Remote: have to communicate across networks
  - Interactive: User gets to “drive” engines in real time, supplying input

Remote Interactive Services with “proxy”

- Proxy is a “catch all” term, meaning some subset of the following functions:
  - Supply network information to client for connection to engines
  - Secure connection (authentication, encryption) between client and engines
  - Accepting connection from client
  - Multiplexing communication between client and engines
  - Allocating nodes to run engines
  - Starting/monitoring/stopping engines
  - Remote data access and checkpointing



- Public IP Control Communication (TCP/IP)
- Private IP Control Communication (TCP/IP)
- Data Communication (Infiniband, Myrinet)

Remote, Interactive Service Examples

	MATLAB DCT/DCE	Star-P	Eclipse PTP	iPython
Client	MATLAB Distributed Computing Toolkit	MATLAB with Star-P Toolkit	Eclipse PTP	iPython Remote Controller object
Proxy	Scheduler/Job Manager smpd	Admin Server HPC Server	PPT ORTE Proxy	ipcontroller
Engines	MATLAB DCE	Star-P Processors	gdb processes	ipengine

ParaM - HPC CSIDE Distribution

http://www.bluecollarcomputing.org

- Paramake - installer for bcMPI
  - UNIX tools, OpenMPI, bcMPI library, bcMPI toolbox, examples, GNU Octave (with vendor BLAS and fftw) if desired
- bcMPI features
  - Runs on UNIX: tested on Linux, NetBSD, MacOS X
  - API “reasonably compatible” with MatlabMPI
    - bcMPI tags are numeric, MatlabMPI alphanumeric
  - Broadcast, barrier, reduce operations
  - bcMPI supports synchronous or asynchronous sends
    - MPI_Buffer_attach, MPI_Buffer_detatch, MPI_Probe
  - MPI communicator support (new in v1.1)
  - Supports many MATLAB data types, but no sparse support
  - MATLAB-style help for commands

ParaM - HPC CSIDE Distribution

http://www.bluecollarcomputing.org

- pMatlab PGAS library from MIT-LL
- Integration with standard HPC environments and tools
  - Job control with PBS, LSF (new in v1.1)
- Remote, interactive service possible, but not planned
- ParaM cluster installations:
  - At OSC: Pentium 4 with Infiniband, AMD Opteron with Infinband (2), Itanium with Myrinet
    - OSC system administrators asked for ParaM since its process launch was integrated with batch startup/shutdown
  - At ASC MSRC: AMD Opteron with Infiniband
  - At ARL MSRC: AMD Opteron with Myrinet

Software Engineering Challenges of HPC CSIDEs

- User community immune to software engineering
  - CSIDE users don’t think of themselves as “programmers”
  - ParaM developed with source control, regression tests, documentation. Users did not understand those concepts!
- Environment mismatch
  - Users: Windows and Web
  - HPC: UNIX and CLI
- Integration is key: standalone solutions to “run parallel MATLAB” not enough: need domain libraries, graphics
- Underestimated user’s ability to do system installation - complete package management system is required

Conclusions

- CSIDE benefits appealing to users
- Problems of moving CSIDEs to HPC systems
  - Technical problems: job launch, control, remote communication
  - Software engineering problems: getting CSIDE users to understand the environment
- ParaM is a first step
  - Job control, interprocessor communication, installer
  - Remote interactivity, performance transparency, move installer to package manager all required