CPPvm - Parallel Programming in C++

Steffen Görzig
DaimlerChrysler
Research and Technology 3
Software Architecture

Topics

- What is CPPvm?
- Why should I use CPPvm?
- Data transfer
- Comparison
- Performance
- Statistics and applications
- What's next?

What is CPPvm?

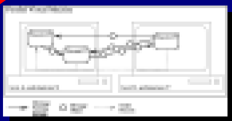
CPPvm (C Plus Plus PVM) is a C++ class library
for message passing built on top of PVM (Parallel Virtual Machine). CPPvm allows to:

- Combine a heterogeneous collection of computers
- Spawn and kill processes dynamically
- Detect failed processes and hosts
- Send/receive C++ objects
- Use distributed C++ objects
- Write your own message C++ classes
- Use standard template library (STL) classes
- Use semaphores
- Use mutual exclusion
- Use CPPvm together with existing PVM software

Why should I use CPPvm?

- It closes the gap between the design of object-oriented parallel programs in C++ and the underlying message passing possibilities of PVM
- Unique message passing features
- Easy-to-use
- Scales from simple examples to complex parallel programs
- Extensive documentation
- Available on many architectures

Data transfer



Explicit message passing:

- Send:
  - blocking
  - non-blocking
- Receive:
  - blocking
  - non-blocking
  - timeout
- Types:
  - CPPvm message classes
  - the standard C++ types bool, char, double, float, int and long as well as constants
  - the standard template library (STL) classes bitset, complex, deque, list, map, multimap, multiset, pair, priority_queue, queue, set, slist, stack, string, valarray, and vector

Distributed Objects:

- Read object from global database
- Write object to global database
- Types:
  - CPPvm message classes

User defined classes for:

- explicit message passing and
- distributed objects.

Comparison

Features	PVM 3.4.3	MPI 2.0	CPPvm 1.4.0	Pvm++ 0.6.0	EasyPvm	Para++ 2.1	OOMPI 1.0.3
Process Handling							
Dynamic Processes	🟩	🟩	🟩	🟩	🟩	🟩	🟥
Dynamic Nodes	🟩	🟧	🟩	🟥	🟥	🟥	🟥
Status	🟩	🟥	🟩	🟩	🟩	🟥	🟥
Status Notification Messages	🟩	🟥	🟩	🟥	🟥	🟥	🟥
Process Synchonization							
Message Passing	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Shared Data	🟩	🟧	🟩	🟥	🟥	🟥	🟥
Mutex	🟥	🟩	🟩	🟥	🟥	🟥	🟥
Semaphores	🟥	🟥	🟩	🟥	🟥	🟥	🟥
Message Passing							
Send Blocking	🟥	🟩	🟩	🟥	🟥	🟥	🟩
Send Non-Blocking	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Receive Blocking	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Receive Non-Blocking	🟩	🟩	🟩	🟧	🟧	🟧	🟩
Receive Timeout	🟩	🟥	🟩	🟩	🟥	🟥	🟥
Forward	🟩	🟥	🟩	🟥	🟥	🟥	🟥
Group Broadcast	🟩	🟩	🟩	🟥	🟥	🟩	🟩
Multicast	🟩	🟥	🟩	🟩	🟥	🟩	🟥
Shared Data							
Mailbox	🟩	🟧	🟥	🟥	🟥	🟥	🟥
Distributed Objects	🟥	🟥	🟩	🟥	🟥	🟥	🟥
Message Control							
Message Passing Context	🟩	🟩	🟩	🟥	🟥	🟩	🟩
Groups	🟩	🟩	🟩	🟥	🟥	🟩	🟩
Communication Topologies	🟥	🟩	🟥	🟥	🟥	🟥	🟩
Message Handler	🟩	🟥	🟥	🟩	🟥	🟥	🟥
Parallel I/O							
Catch Stdout	🟩	🟧	🟩	🟥	🟥	🟩	🟧
Files	🟥	🟩	🟥	🟥	🟥	🟥	🟥
Message Types							
Standard C Types	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Standard C++ Type	🟥	🟩	🟩	🟩	🟥	🟥	🟥
Standard Template Library (STL)	🟥	🟥	🟩	🟩	🟧	🟥	🟥
C++ Templates	🟥	🟥	🟩	🟩	🟥	🟧	🟥
User Defined Types	🟥	🟩	🟩	🟩	🟥	🟥	🟩
Language Support							
C	🟩	🟩	🟥	🟥	🟥	🟥	🟥
Fortran	🟩	🟩	🟥	🟥	🟥	🟥	🟥
C++	🟥	🟩	🟩	🟩	🟩	🟩	🟩
OO/C++ Concepts for Applications							
Inheritance	🟥	🟩	🟩	🟩	🟥	🟥	🟩
Polymorphism	🟥	🟩	🟩	🟩	🟥	🟥	🟥
Overloaded Operators	🟥	🟩	🟩	🟩	🟩	🟩	🟩
Exception Handling	🟥	🟩	🟩	🟥	🟥	🟥	🟩
User Defined Templates	🟥	🟥	🟩	🟩	🟥	🟥	🟥
Streams for Messages	🟥	🟥	🟩	🟥	🟩	🟩	🟩
Namespaces	🟥	🟩	🟥	🟩	🟥	🟥	🟥

WWW statistics (4/99-8/01):

- > 11.000 requests
-