The PVM System: Recent Results & Future Directions

Vaidy Sunderam
Emory University
Atlanta, GA 30322, USA
+1 (404) 727-5926
vss@mathcs.emory.edu

- Scope: PVM evolution, recent results, performance issues, ongoing research, and future directions.

- Outline
  - Introduction and Background
  - Recent Projects
  - Performance Optimization
  - The GDC Layer
  - Concluding Remarks

Background

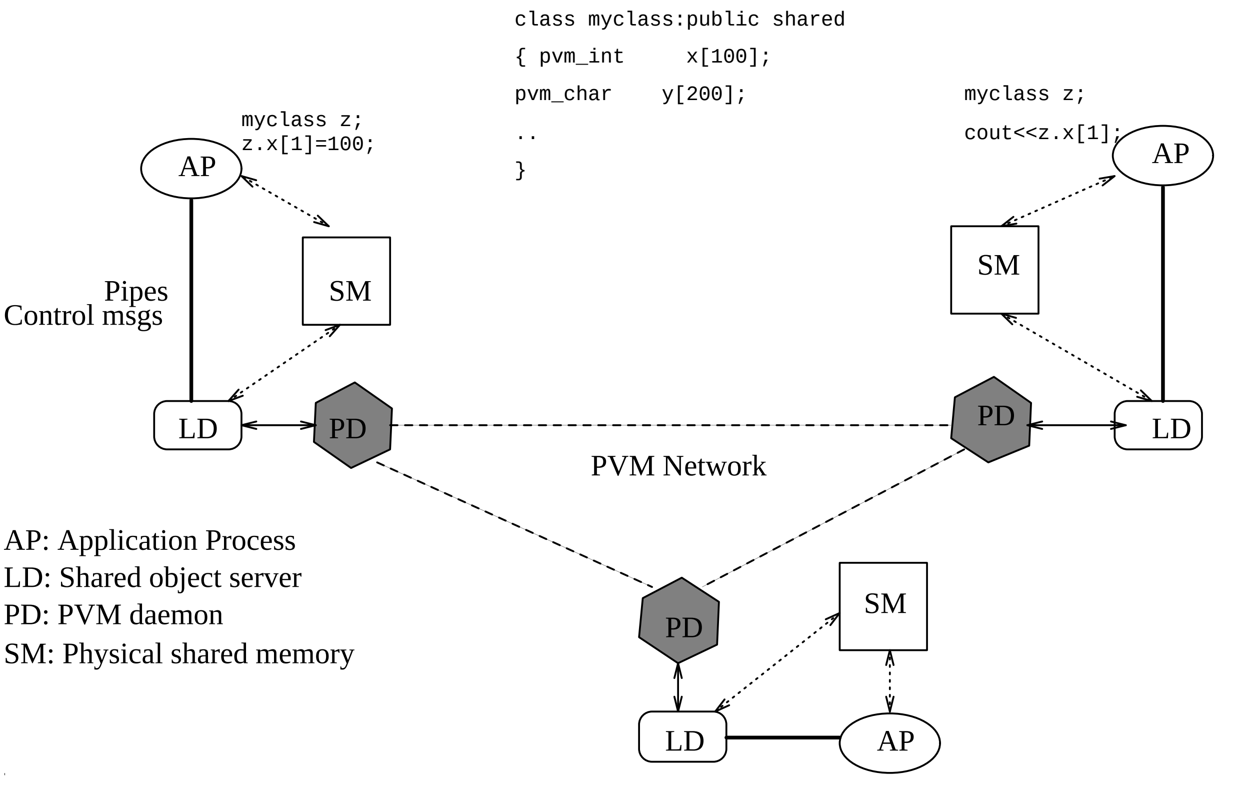
- PVM Evolution
  - 1989 - Version 1.0 : defined computing, operational, and implementation models; included "research" features (DSM, recovery, dist. algs.); proof of concept implementation on several platforms.
  - 1990 - 1.0 + Xpvm : Enhancements, graphical interfaces; viability for scientific supercomputing.
  - 1991 - 2.x release versions : Widespread adoption, large diversity/number of applications; HeNCE development.
  - 1992 - 3.0 and beyond : Syntactic and minor semantic changes to interface; re-engineered implementation; new features (VM reconfiguration, MPP support).

- Project Methodology
  - Experimental research effort (DoE, NSF, NASA, TN) : software is by-product of research investigations.
  - Core release in public domain; research extensions incorporated when succesful and worthwhile.
  - Extensions and enhancements : Closely intertwined with core system - e.g. protocols, new primitives.
  - Auxiliary subsystems : Relatively independent, layered above core system.

Recent Projects

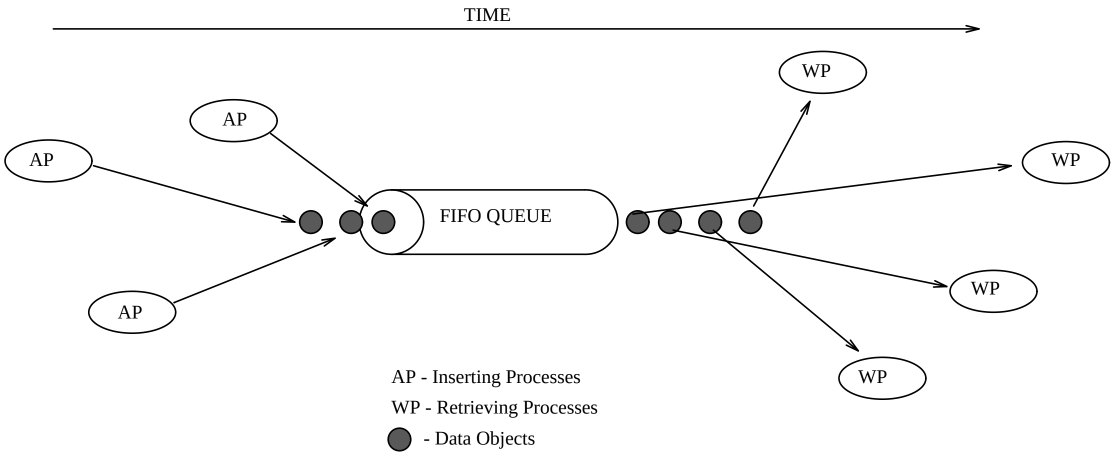
- Multitude of auxiliary projects : performance, debugging, profiling, application-domain libraries, etc.

- Shared Object Programming
  - Virtual Shared Object Space, shared object programming using the C++ language
  - Transparent access to shared objects, consistency
  - Facilities for partitioning, workload distribution
  - Performance comparable to native message passing implementations
  - Architectural Overview



This page used for formatting purposes only

– Queues for workload management




```
myclass     obj;
fqueue      fifo;
fifo.set_id(1)
fifo<<obj;  //insertion
```

```
  myclass   obj;
  fqueue    fifo;
fifo.set_id(1);
  fifo>>obj //retrieval
```

– Preliminary performance data

# Procs	MP Matmul	SO Matmul	FIFO Matmul
1	930	930	930
2	511	521	487
4	270	282	166
8	175	177	105
16	112	101	95

- Status
  - Alpha version available; supports aggregates, locking.
  - Future work - proactive object location, signalling, multithreaded.

Performance Related Research

- Primitive communication speeds : Ethernet vs. FDDI

- Data Compression to Lower Communication Costs
  - Advantages : Shorter messages lead to (1) Increased throughput, *and* (2) lowered end-to-end latency (fewer packets transmitted).
  - Drawbacks : (1) Cost of compression; (2) Generally, numeric data is not compressible!

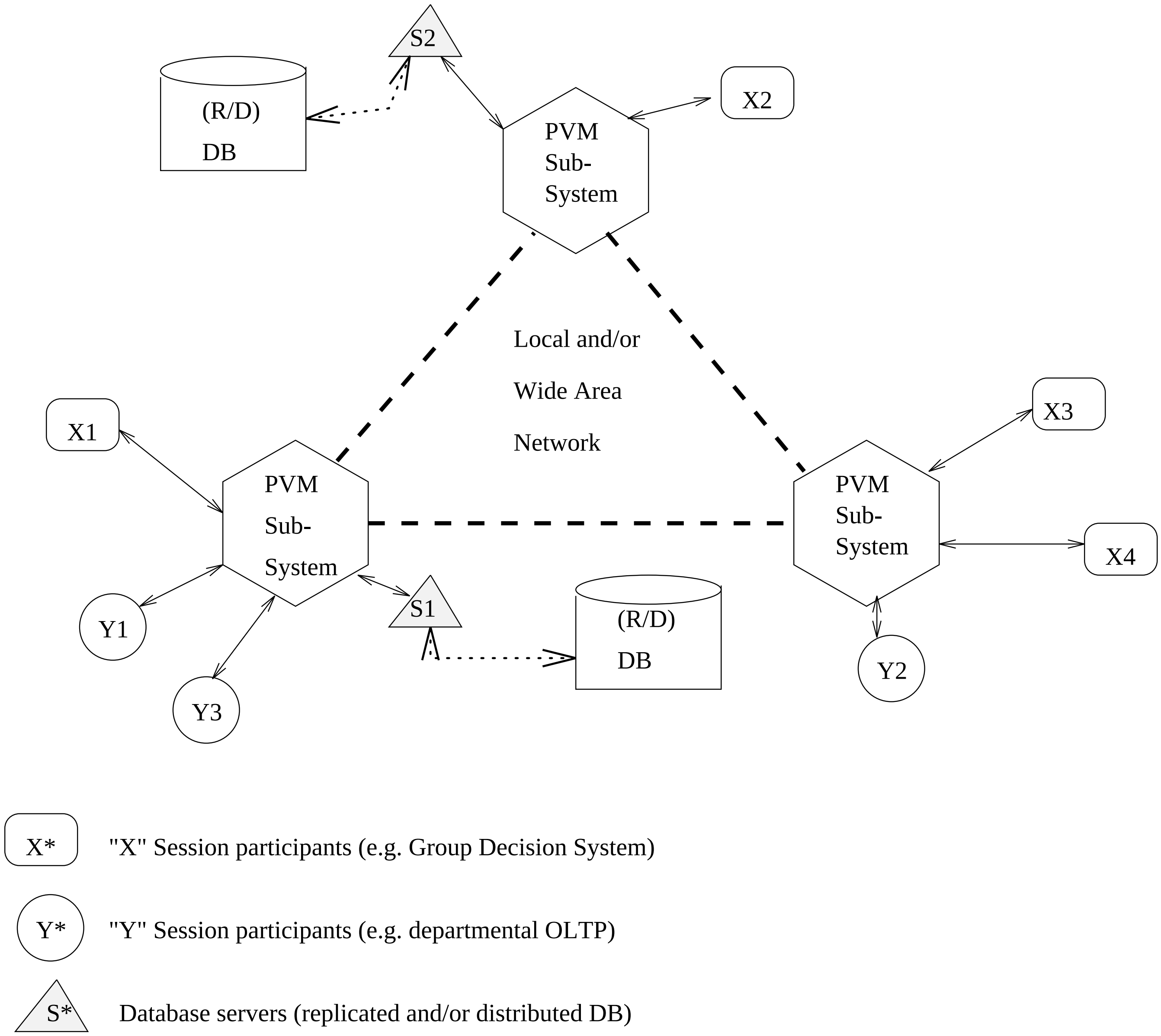
- Techniques to overcome drawbacks
  - Floats and doubles
    + Avoid packing sign bit if possible. Pack exponent and significand using the most bits used by largest.
    + Avoid packing sign bit if possible. Pack exponent in the most bits used by anyone. Pack significand in exactly the number of bits necessary.
    + Avoid packing sign bit if possible. Pack exponent and significand in exactly the number of bits necessary.
  - Integers
    + Pack in as many bits as the largest integer requires.
    + Pack each integer using only needed bits (5-bit header).
    + Pack each integer using only needed bytes (2-bit header).

- Eliminating protocol overheads
  - Large proportion of PVM use is on local networks
  - Alternative protocol interfaces can eliminate overheads of network and transport layer
  - Direct operation over DLL feasible in many situations; permits access to network broadcast/multicast, optimal packet size etc.
  - Latency improvement of upto 50% observed in test implementation.

The GDC Layer

- Support for generalized distributed computing and commercial applications
  - Rationale and Motivations
    - Groupware applications (calendar management, teleconferencing).
    - Group decision support systems
    - Heterogeneous or multi-database access
    - Database applications: (1) complex queries against massive (replicated and/or distributed) databases; (2) High speed OLTP.
  - Infrastructural requirements
    - Parallel I/O
    - Locking and synchronization
    - Data replication, fault tolerance
    - Client/Server paradigm
    - Transaction processing support

- GDC Architectural Model



Architectural Overview of GDC Layer

This page used for formatting purposes only

- Sessions and Contexts
  - Multiple, possibly intersecting, sessions or groups
    + Different consistency and FT semantics as appropriate
    + **`pvm_checkin("session_name",`** *<auth params>*`,` *<semantics>*`, . . .);`
    + Returns *session_id* and *participant_id*.
    + A process may check into multiple sessions
    + **`pvm_checkout(`***<session_id>***`);`**

- Parallel I/O in PVM
  - Extension of Unix filesystem semantics
  - Different levels of shadowing, consistency semantics
  - Basic operations
    + **`pvm_open("file[@machine]",`** *<options>***`);`**
    + Options: conversion, compression, encryption, shadowing, access mode
    + Returns *pvm_fd* handle for subsequent IO.

- `pvm_write(pvm_fd, dp, nitems, . .);`
- `pvm_read(pvm_fd, dp, nitems, . .);`
- Atomic **`seek`** incorporated in read/write.
- **`pvm_close(pvm_fd,`** <*flags*>**`);`**
- Flags for IO include timeout component
- Shadowing: Multiple copies of parallel file; replicated operations; strong or weak ("best effort").
- Serialization: automatic, depending on modes.

- Locking and Synchronization
  - Abstract locking facility, with user-named locks.
  - System support for "persistence", deadlock avoidance, etc.
  - Locks (and resources) named by <*string, number*> tuple.
    - `pvm_lock("lockname", locknum, flags);`
    - Safety and timeout options provided.
    - Example convention for record-level locking:

      `pvm_lock("filename", recnum, AUTO);`

- Client/Server Infrastructure
  - Application-specific services for distributed access
  - Location transparency, data conversion, replicated and/or shadow services for load balancing/failure resilience.
  - Simplified client and server development:
    + **pvm_export("service",** *<procname>*, *<msgtype | arglist>*, **flags);**
    + **pvm_invoke("service", dataptr, TYPE | NARGS, flags);**

- Transaction Processing Support
  - Provide qualified ACID properties
  - Performance and response dependent on nesting and conflicting accesses
  - Ideally suited for "shared nothing" databases, high degree of inter-transaction parallelism.
  - Inter-DML parallelism may be implemented using PVM subsystem and other OS facilities (e.g. threads).

– Straightforward, concise, interface:

+
```
pvm_bt("trans name", number);
        . . .
        pvm_bt("nested", num);
          . . .
        pvm_et();
        if (. . .) pvm_abort();
pvm_et();
```

– Support for recovery, temporary rollback, logging.

– Semantics for message passing within transaction scope are presently unresolved.

- Implementation strategies for GDC

  – Files: Combination of NFS, message-based IO schemes.

  – Other Facilities:

    + Distributed mutual exclusion (FPP, quorum)

    + Distributed state-maintenance algorithms

    + Fast multicast/broadcast & synchronization protocols

Preliminary Performance Results

- Performance of core primitives

No. of parti-cipants	Message Size (bytes)				
	1k	4k	64k	256k	1M
2	2.9	8.2	77.1	312.2	1246.9
4	11.5	19.1	85.5	316.4	1277.2
8	20.5	27.3	92.7	323.9	1293.7
16	38.1	45.6	114.8	351.6	1383.8

Data transfer times in milliseconds

No. of shadow-files	Read/Write Size (bytes)				
	1k	4k	64k	256k	1M
0	2.1	9.1	82.6	343.2	1519.3
1	4.1	15.3	112.6	368.7	1611.8
2	5.9	20.6	138.9	391.0	1689.2

Parallel IO times with synchronous shadowing

No. of parti-cipants	Message Size (R&R; bytes)				
	8	64	128	512	1024
2	4.7	6.6	10.8	15.1	24.3
4	9.1	14.4	18.2	22.8	33.6
8	22.2	25.8	31.4	42.8	63.9
16	39.3	42.8	73.7	96.5	108.2

Times (milliseconds) for "Client/Server" exchanges

Distributed Mutual Exclusion in GDC					
Method	2 Node Group	4 Node Group	8 Node Group	12 Node Group	24 Node Group
VPS	1.2	1.2	1.2	1.2	1.2
SQ	3.2	3.2	13.4	13.4	14.0
MC	3.2	6.4	13.4	28.0	30.5

KEY:
VPS - Variable Primary Site
SQ - Structured Quorum
MC - Majority Consensus

Locking Performance (Milliseconds)

Concluding Remarks

- PVM and Auxiliary Systems
  - Generalized, heterogeneous concurrent computing in networked environments
  - Good to excellent performance and functionality
  - Widespread adoption, impact on HP computing
  - Experimental research resulting in usable technology

- Ongoing work
  - Performance enhancements, new communication architectures
  - Generalized distributed computing facilities
  - Load balancing, partitioning tools, object-model support
  - Graphical visualization and debugging interface
  - Long term goal : "Distributed Object Threads" as a unified concurrent, paralle, and distributed computing paradigm.