Why would I want to `reg`? Customizing PVM to suit your environment

Jim Pruyne

Computer Sciences Department
University of Wisconsin – Madison
`pruyne@cs.wisc.edu`
`http://www.cs.wisc.edu/~condor`

Getting our Hooks into PVM

- *Condor* - a distributed batch system
  - ⋆ Schedules user jobs onto machines (user doesn't know which machines ahead of time)
  - ⋆ Provides its own means of accessing new machines and starting processes
  - ⋆ Applications need services not present in PVM
- *Paradyn* - a performance debugger
  - ⋆ Must create user processes in order to perform instrumentation

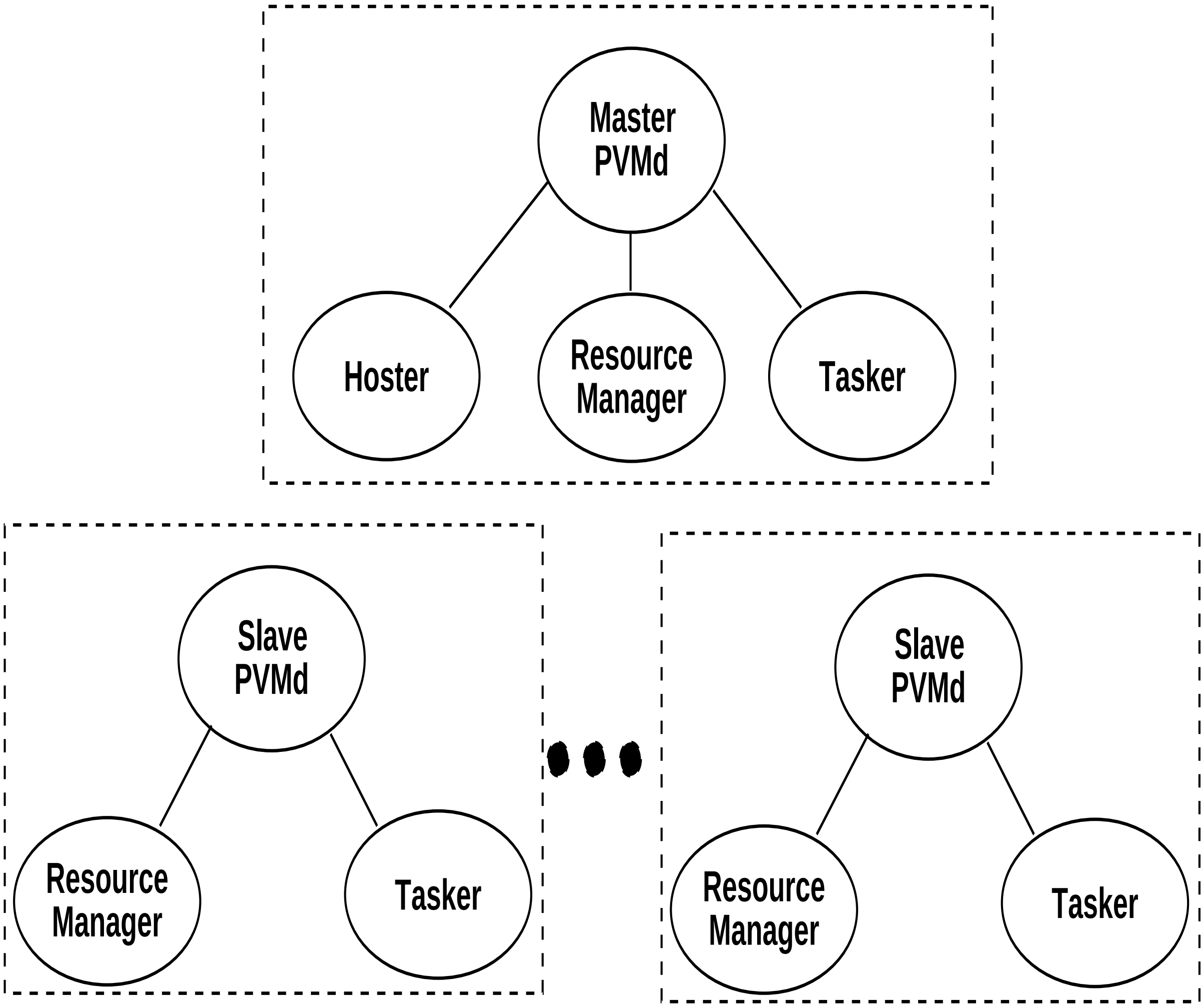
Reasons for Customization

- PVM's built-in policies are not applicable or sufficient for your environment
  - ⋆ Different process placement policies
  - ⋆ Different interpretation of lib. PVM parameters
  - ⋆ New API functions
- PVM's mechanisms may not be usable
  - ⋆ `rsh` may be unavailable
  - ⋆ Process creation may be different

The Customization Hooks

- `pvm_reg_rm()` - Intercept and handle all resource management related PVM calls made by an application
  - ⋆ Arbitrary collection of processes
- `pvm_reg_hoster()` - Assume responsibility for accessing and starting PVM daemon on new machines when they are added
  - ⋆ Only at the master PVMd
- `pvm_reg_tasker()` - Assume responsibility for process creation on a machine (rather than the PVMd)
  - ⋆ One or none per host
- All protocol definitions related to the hooks are defined in `pvmsdpro.h`

A Fully Populated PVM



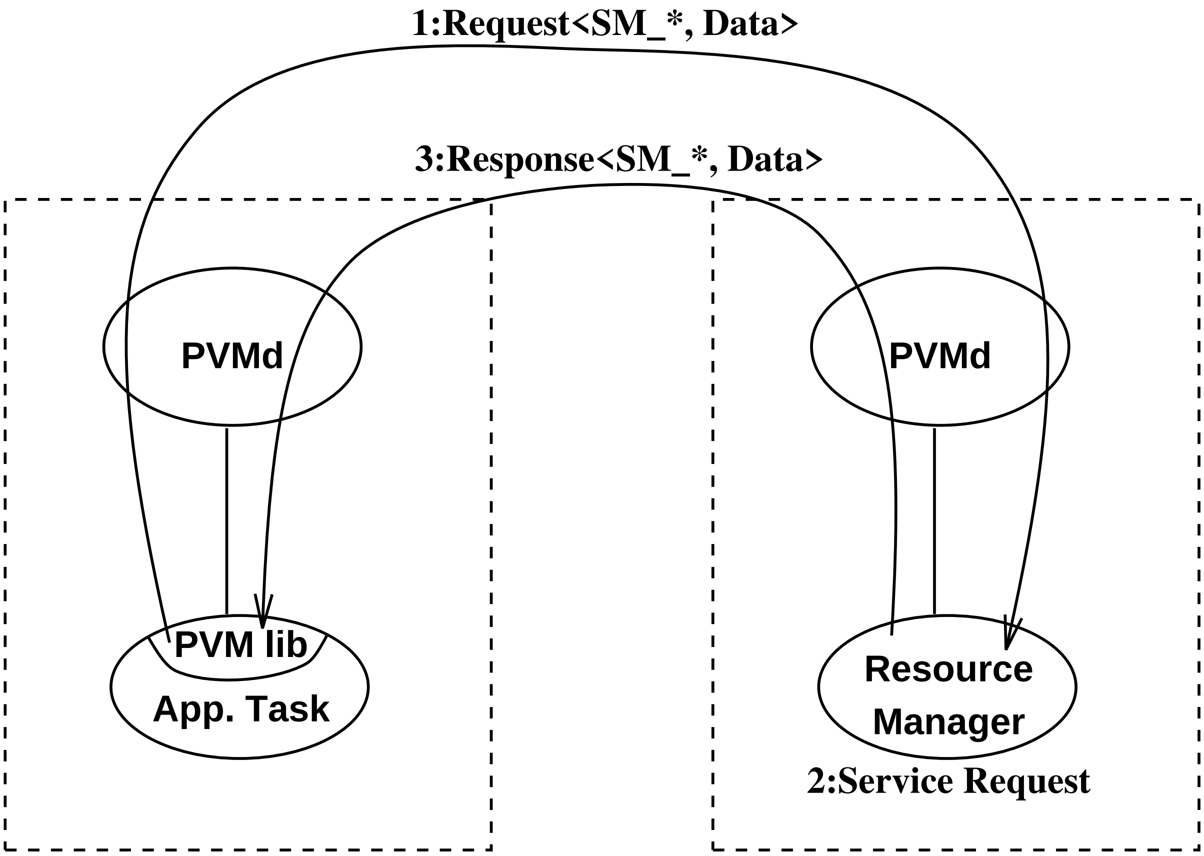
Resource Manager Overview (pvm_reg_rm())

- One RM task assigned to every application task
  - ⋆ Cannot be changed after start-up
  - ⋆ PVM library sends all RM service requests to the RM task via a message
- One RM task assigned to every PVM daemon
  - ⋆ App. tasks started by a daemon inherit the daemon's RM task
  - ⋆ Slave PVMd's inherit the master's RM task when started, but can change RM tasks later (`SM_HANDOFF`)

RM Overview (cont.)

- Multiple RM tasks in one PVM, arbitrary topology
  - ⋆ Scalability
  - ⋆ Locality
- Basic RM services are: `pvm_spawn()`, `pvm_tasks()`, `pvm_config()`, `pvm_addhost()`, `pvm_delhost()`, `pvm_notify()`
- RM services extensible
  - ⋆ *Pro:* New services can be defined which suit your environment
  - ⋆ *Con:* Ties an application to a particular RM

Generic RM Service Request Protocol

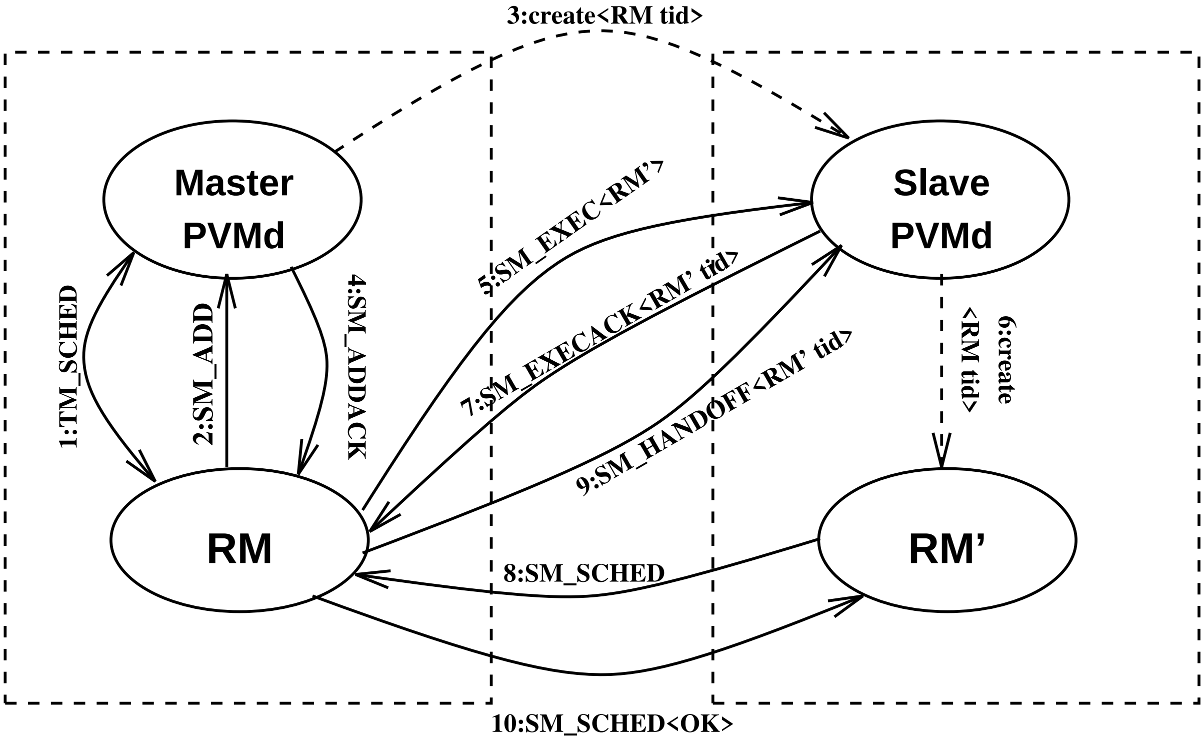


- Request Msg. generated and response processed by `lpvmgen.c`
  - ⋆ Code is the best guide to msg. formats

Servicing RM Requests

- RM Task(s) must maintain state information
- Daemons will start processes (`SM_EXEC`) and add hosts (`SM_ADD` only at master) only for their RM task
  - ⋆ Invokes tasker and hoster at lower layer
  - ⋆ `pvm_setopt(PvmResvTids, 1)` to access
  - ⋆ Automatic notification on task exit (`SM_TASKX`) and host exit (`SM_HOSTX`) by daemons
- RM tasks may cooperate in any way

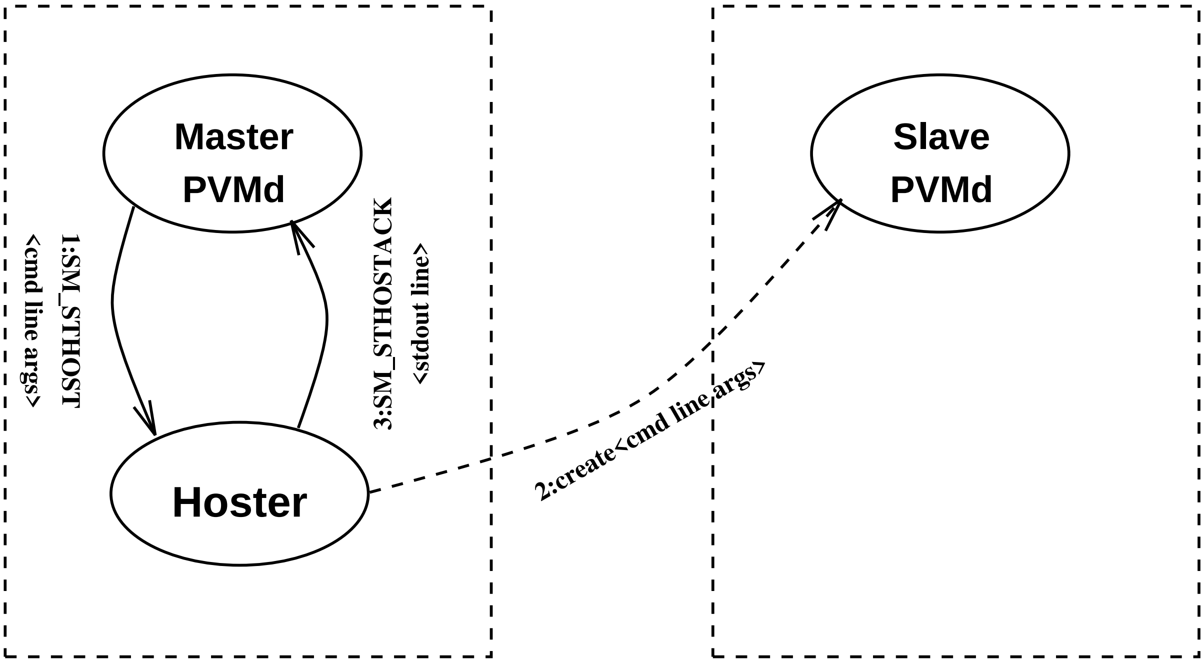
Evolution of a multi-RM environment



Hoster Overview
(pvm_reg_hoster())

- Provides access to machines, and starts a PVMd
  - ⋆ Replaces the `rsh` or `rexec` in the master PVMd
- Hoster must be registered on the same machine as the master PVMd
  - ⋆ Host table update protocol driven by master
- Not responsible for detecting host failure
- Example code in hoster directory of PVM distribution

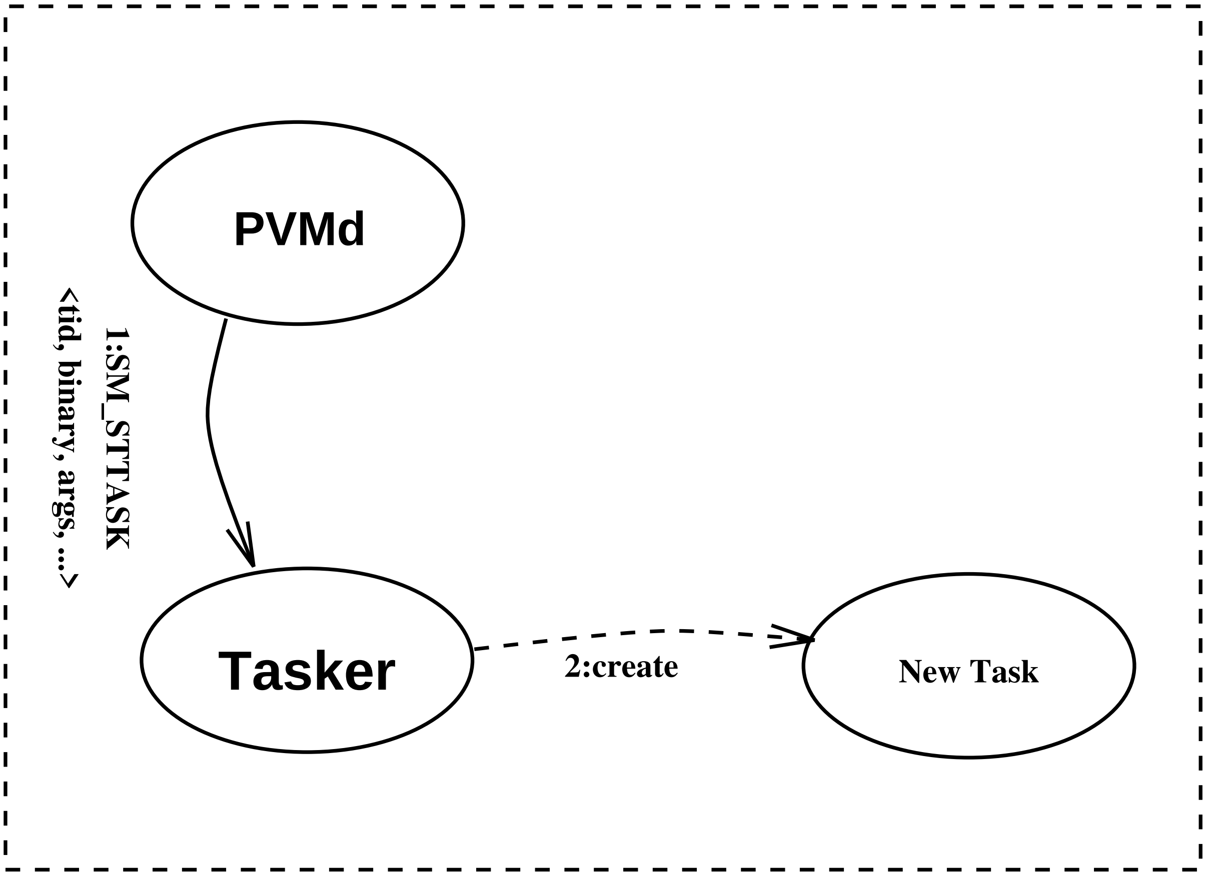
Host Add Protocol



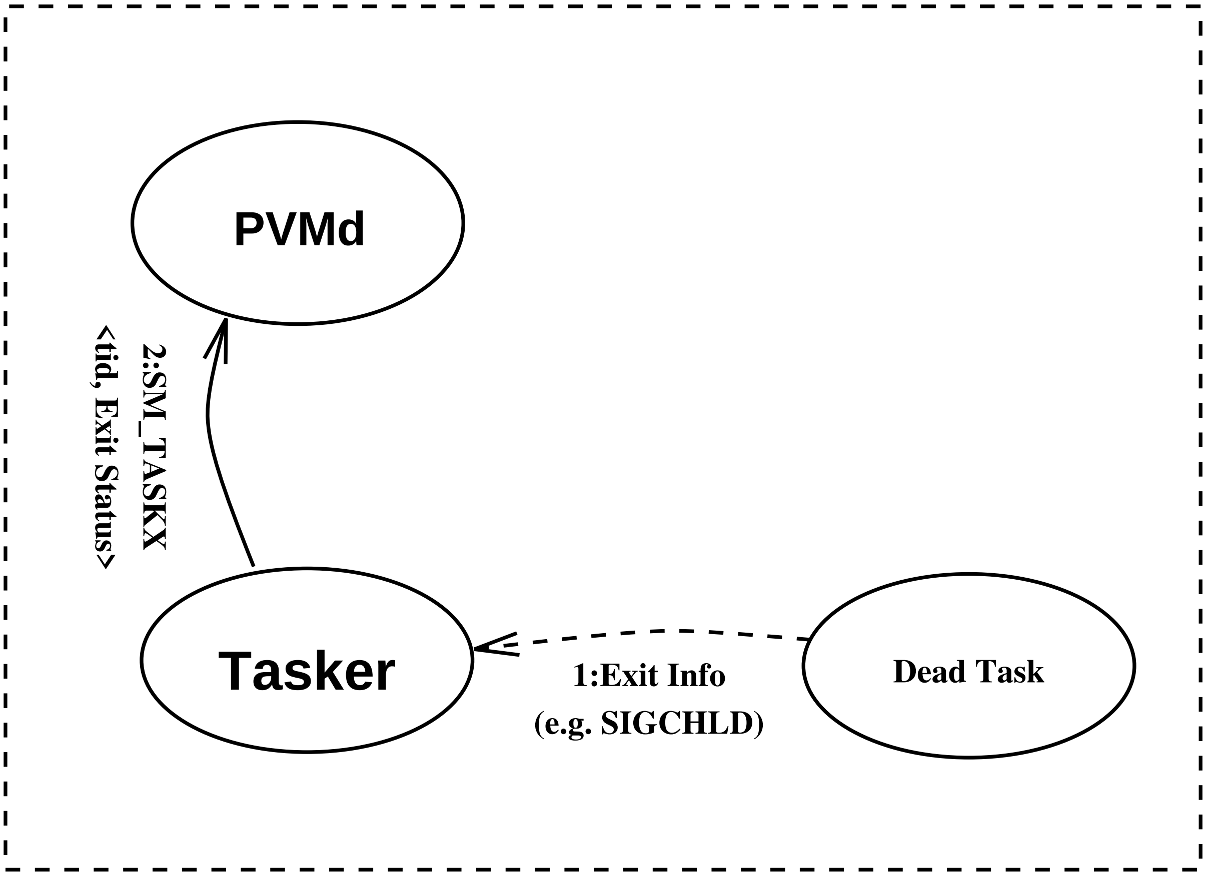
Tasker Overview (pvm_reg_tasker())

- Responsible for creating processes and signalling daemon when processes exit
  - ⋆ Replaces Unix `fork()/exec()` built into daemons
  - ⋆ Must use `pvm_setopt(PvmResvTids, 1)` to inform daemon of process exit (via `SM_TASKX`)
- Tasker must be on same machine as daemon it attaches to
- Uses: debugging, accounting, performance monitoring, ...
- Example code in tasker directory of PVM distribution

Task Creation Protocol



Task Exit Protocol



Summary

- Customization allows PVM to be used in many environments
  - ⋆ Addresses OS differences, RM policies, debugging, accounting, security, ...
- Default behavior for everything already built into PVM
  - ⋆ Build only what you need
  - ⋆ Start from the examples
- Designed for extensibility