

Parallel virtual libraries: a system for resource management

Andrea Clematis

IMA, CNR

Via De Marini, 6 - 16149 Genova, Italy

`clematis@ima.ge.cnr.it`

Gabriella Doderò, Vittoria Gianuzzi

DISI, Università di Genova

Via Dodecaneso, 35 - 16146 Genova, Italy

`{doderò,gianuzzi}@disi.unige.it`

This research is partially funded within a Coordinated Project of CNR, and aims at supporting solutions for computationally intensive problems on Networks of Workstations. Such networks usually are highly non-homogeneous, thus requiring a specialized competence in scaling and locating parallel applications, in order to maintain load balancing.

There are few environments supporting easy-to-use access to heterogeneous distributed systems, where mathematical libraries are available and access to resources is transparent. One such system is NetSolve, developed by Jack Dongarra and his group, which provides an interactive interface as well as another for programming languages like Fortran77 and C. Load balancing and fault tolerance are managed by NetSolve, however no special support is provided for parallel libraries.

The toolset which is being designed and developed, PINCO (Programming environment for Network COmputing), provides a safe user interface that activates the required parallel functions on those workstations and clusters that at each moment may guarantee the best performance. It thus supports the creation and use of Parallel Virtual Libraries, an expanding application field for parallel and distributed computing.

Pinco consists on three components: the Job Scheduler, the Resource and Task Manager, and the Application Interface. At present, a subset of functionalities has already been implemented; the above decomposition shall allow experiencing with different job and resource management policies, by substituting one component only.

The existing tools are at present:

- ★ PPE, which allows to compile and allocate both Pinco and the parallel applications, and to perform off-line evaluation of computing power available at nodes of different architectures;
- ★ PRTM, the distributed Pinco Resource and Task Manager, which periodically evaluates workload at each node, processes application requests for

computing power, and eventually activates parallel tasks.

Task synchronization and process migration have already been included in Pinco design, but they are not yet available at present.

PPE evaluation of available computing power on the heterogeneous network takes place as an off-line process. It consists on benchmarking CPU execution times at each node for a selected program mix including both floating and fixed point computationally intensive packages. The unit of performance is the measured performance of the benchmark running on a Pentium 133 Mhz under Linux. The heterogeneous network where PPE is presently running includes various models of SparcStations, UltraSparc, Silicon Graphics, HP9000 and others.

In lack of a scheduler, a simplified API provides only a few calls to spawn, monitor and control tasks under PRTM control. The burden of tasks allocation is left under application control, in order to experience with different scheduling policies, before embedding them inside the system. SPMD or Master/Worker applications, where the same code for tasks (except for the Master Task) is executed on all nodes, are the programming paradigms which better match Pinco features, and which can already be executed within present toolset limitations.

PRTM is not just a task which may be called "the Allocator". Such a functionality is distributed across the network, yet maintaining a centralized information collector, that is the Resource Manager. The latter is a process running on the master host node of the network, where information sent by Local Load Monitors is kept, and used to take decisions about process allocation to the network.

At present allocation decisions result from negotiation between PRTM and the user process, as the choice suggested by the Resource Monitor are made visible and modifiable, if needed, by the user. As far as experience in this decision making process is collected, it is expected to include "smart" allocation policies in the Resource Monitor, thus making user supervision no longer needed.

Information on available computing power can be classified as:

- ★ static info: initialized as collected by PPE;
- ★ dynamic info: computed and periodically updated by the Local Load Monitor.

Each Local Load Monitor runs on a different Pinco host computer. The percentage of CPU free is computed every 5 seconds by issuing a local system call, such as the command `ps`. This value is actually sent only if different for more than 10% from the one previously sent to the Resource Manager.

When the Resource manager receives a message from the Local Load Monitor at node `x`, it updates the so called `load_table` where dynamic and static network load parameters are kept. The static "weight" of node `x` is multiplied by the percentage of CPU free, thus giving the available power on node `x` for Pinco requests. This value is still expressed in "Pentium 133

Units”, truncated to the first decimal digit for simplicity.

Allocation requests from user processes are based on a data structure called `grant_record`. It is filled by the Resource Manager and passed to the user process by means of a two phase protocol:

★ in the first phase, the user process issues a request expressed by the call

```
err = pinco_grant(requestedpower, grant_record);
```

★ the user process may check compliance of the proposed allocation to possible additional constraints and then issues the activation request:

```
err = pinco_spawn(executablefile, grant_record);
```

The function `pinco_grant` fills a list of pairs

```
(processor_name, local_granted_power),
```

such that the sum of all `local_granted_power` fields matches the request or best approximates it by default if impossible.

Loads corresponding to `local_granted_power` are reserved for the process that invoked the `pinco_grant`, but parallel processes are spawned only when a `pinco_spawn` is invoked by the user process. Actual number of parallel tasks per processor is specified, by adding a third value in the node list of the `grant_record`. The user process may also update the `grant_record`, before invoking `pinco_spawn`, in order to use just a subset of available resources, thus leaving residual computing power to other applications.