

Load Balancing With Migration Based on Synchronizers in PS-GRADE Graphical Tool

M. Tudruj^{1,2}, J. Borkowski¹, D. Kopański¹

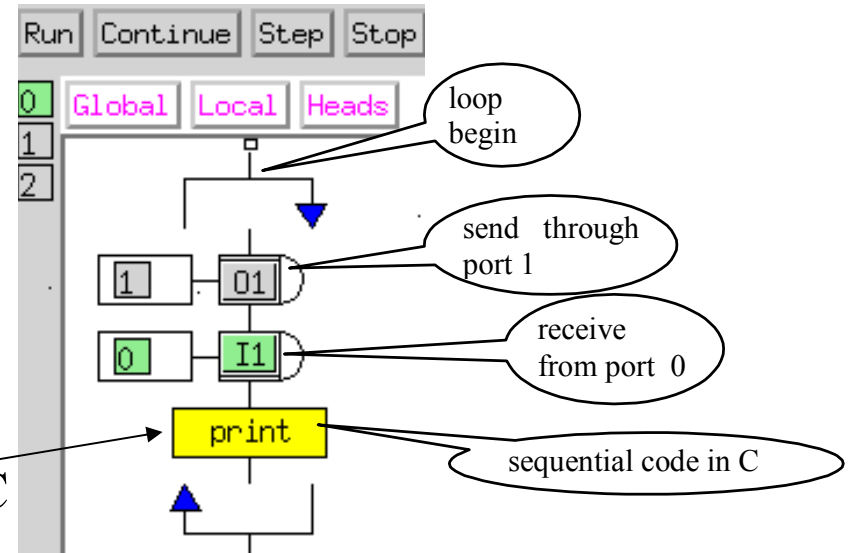
¹ Polish-Japanese Institute of Information Technology, 86
Koszykowa Str., 02-008 Warsaw, Poland

² Institute of Computer Science, Polish Academy of Sciences, 21
Ordona Str., 01-237 Warsaw, Poland
{tudruj, janb, damian }@pjwstk.edu.pl

P-Grade system

A complete graphical programming environment for developing message passing applications from Parallel and Distributed Systems Laboratory of the SZTAKI Institute of Hungarian Academy of Sciences

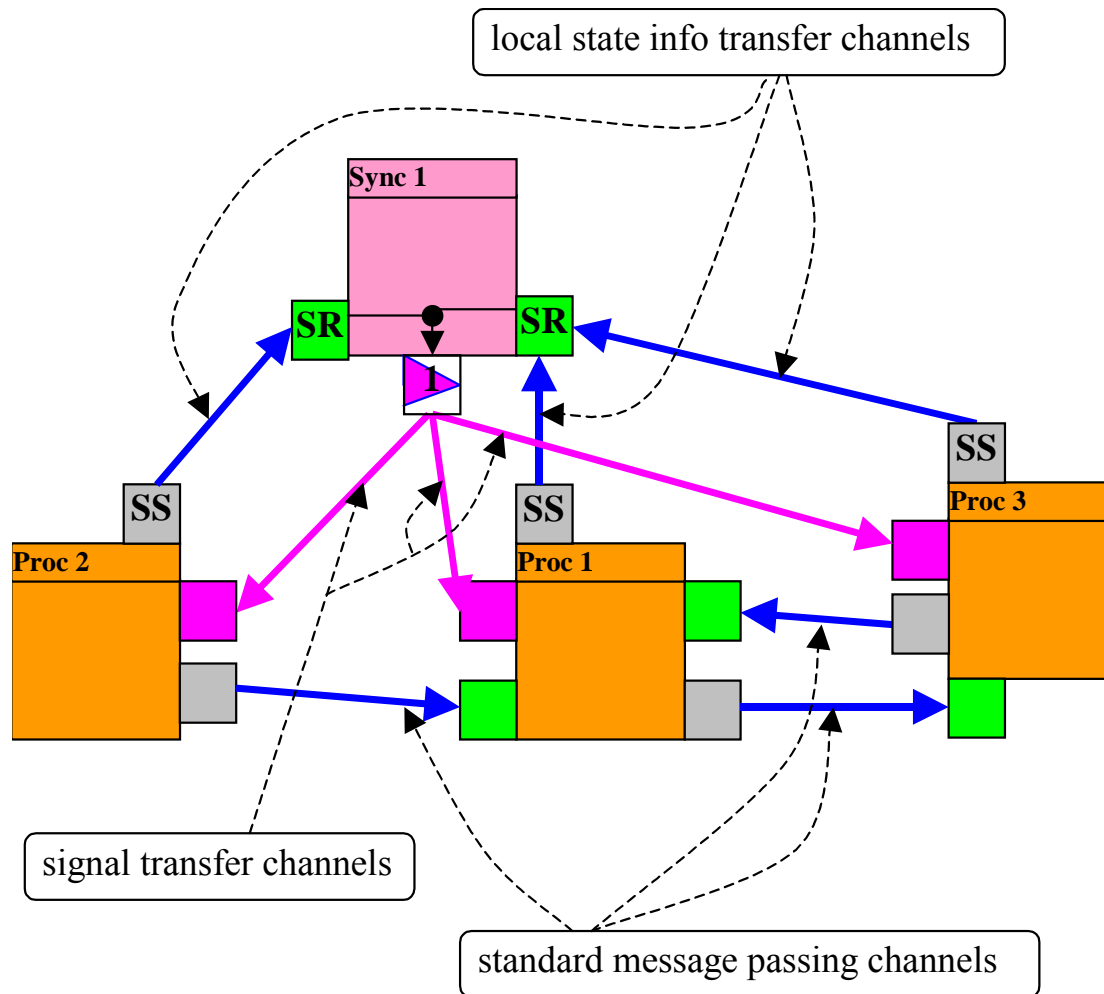
- Application level specifies processes and their interconnections
- Process level defines control flow diagram of a process
- Text level is used to enter sequential C code into elements of a flow diagram



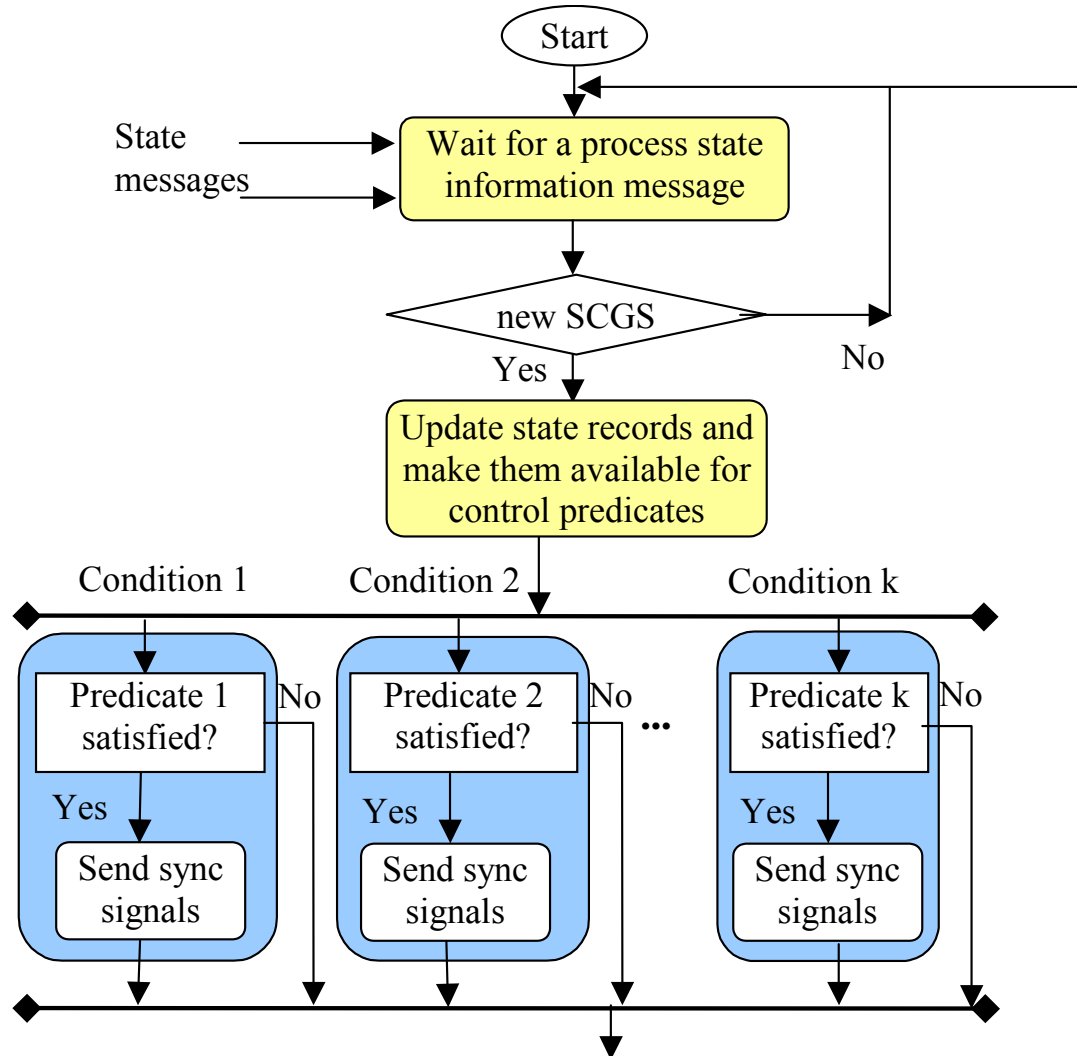
PS-Grade – synchronization oriented

- Process report their states to synchronizer processes
- Synchronizers observe application global states and evaluate predicates on them
- Predicate evaluation can cause sending control signals to processes
- The signals can activate assigned procedure, or they can cancel current computations

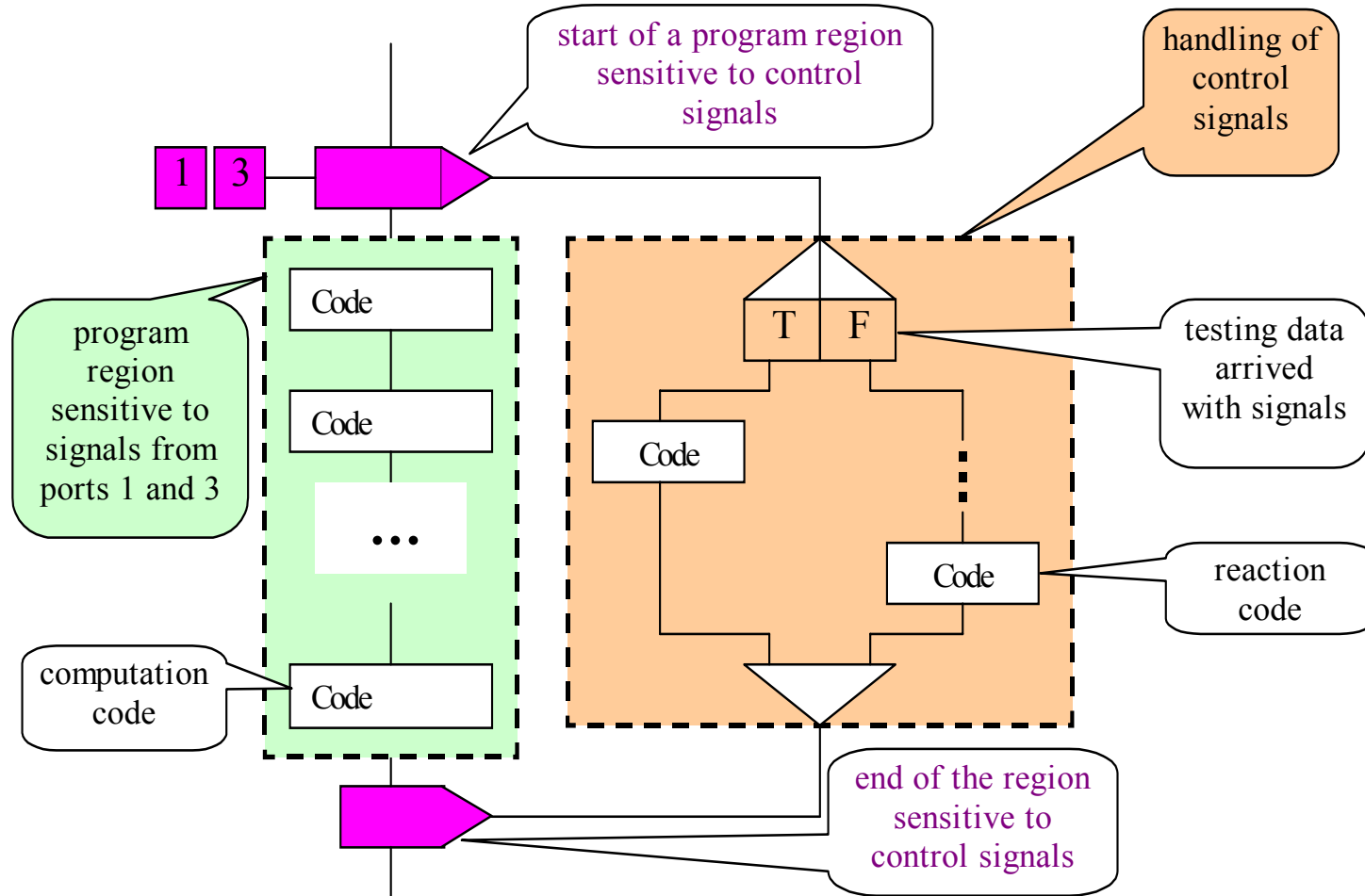
PS-Grade – application level



PS-Grade – synchronizer



PS-Grade – process control flow



PS-Grade – reaction on signals

- Control signals are received only in signal sensitive regions
- Signals are handled immediately
 - Current computations are suspended to handle a signal, then they are resumed,
 - Current computations are suspended to handle a signal, then the rest of the region is omitted and program continues after the region
- Signal handling can be temporarily blocked (e.g. when executing non-reentrant procedures)

Load balancing in PS-Grade

- Ready infrastructure to collect load information
- Easy formulation of load balancing strategies in the form of global predicates
- Existing mechanism for asynchronous notification

Load balancing in PS-Grade - methods

- Data redistribution
- Execution migration
- Process migration with user-controlled process state transfer
- Synchronizer-driven using external libraries
- Completely realized using third-party tools

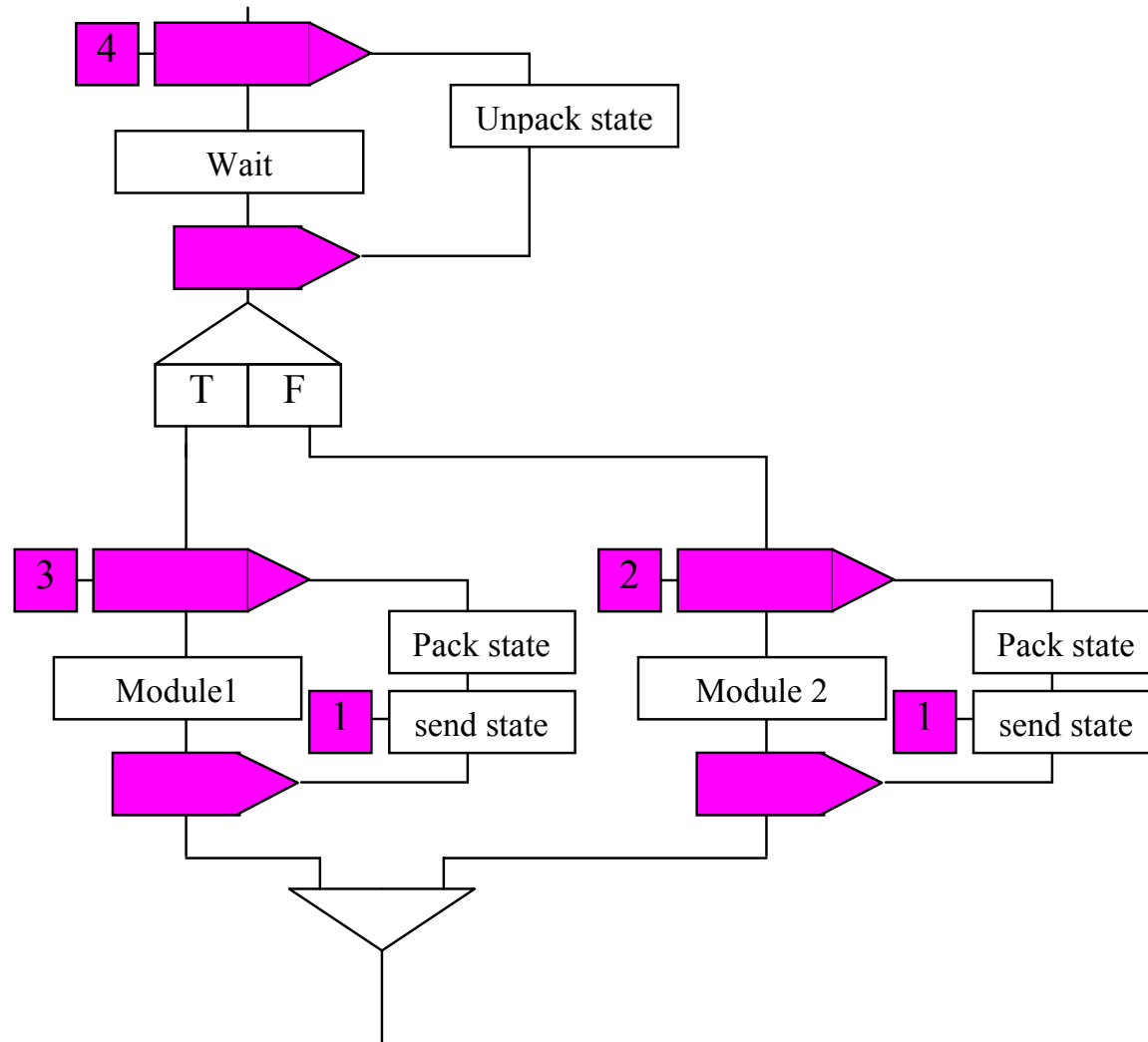
Data redistribution

- A user-supplied signal handler is responsible for packing/unpacking part of process data
- Synchronizer decides which process should transfer part of its data and which one should get them
- Implemented in parallel Travelling Salesman Problem

Execution migration

- Processes are composed of modules
- Modules execution can be transferred from node to node
- User code within the processes reports load level to a synchronizer
- The synchronizer decides which module should be run and where

Execution migration – cont.



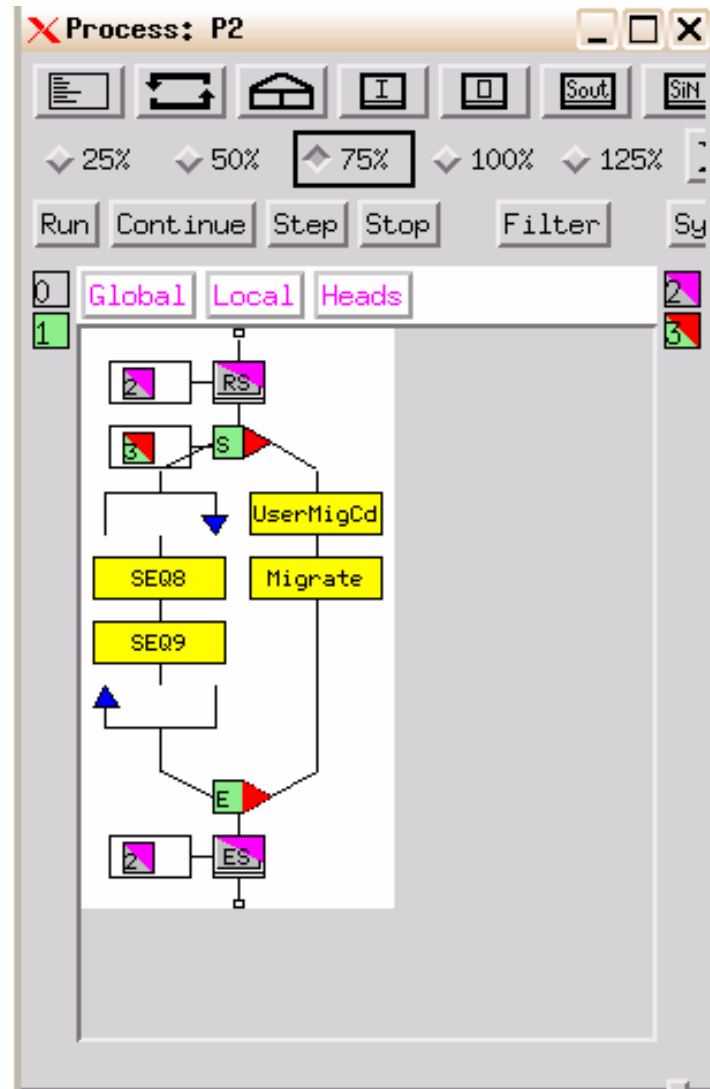
Migration with user-controlled process state transfer

- Special system load monitoring processes introduced, they report system load to synchronizer
- Application processes can pack/unpack their states and save/load them to/from network disk
- Migration:
 - Synchronizer decides about migration
 - Synchronizer notifies a process to save its state, and spawns a new copy of the process at another location,
 - The migrated process notifies system processes about its new location
 - The old copy continues as a message forwarder.

Synchronizer-driven migration using external libraries

- DPVM library facilitates process migration
- Checkpointing, process restarting, message redirection is done transparently by DPVM
- Load monitoring processes report system load to synchronizer
- Synchronizer decides about migration and sends a control signal to a process
- Application process reacts on the signal by calling DPVM function `pvm_move ()`
- Special care must be taken to deal with shared memory and UNIX signals – not supported by DPVM, but used by PS-GRADE

Synchronizer-driven migration process template



Migration controlled by external tools

- DPVM includes system load monitoring and a built-in migration strategy
- DPVM can totally control process migration
- Simpler – fully done by the library
- No modification in application process code necessary
- Not flexible – predefined load measures and migration strategy

Conclusions

- PS-Grade constitutes a good framework for implementing load balancing strategies
- A number of load balancing methods can be realized, from fully user programmed to fully handled by a system library
- User-programmed load monitoring and control predicates let us define and deploy easily a variety of load balancing strategies