Consistent Checkpoints of PVM Applications

G. Stellner

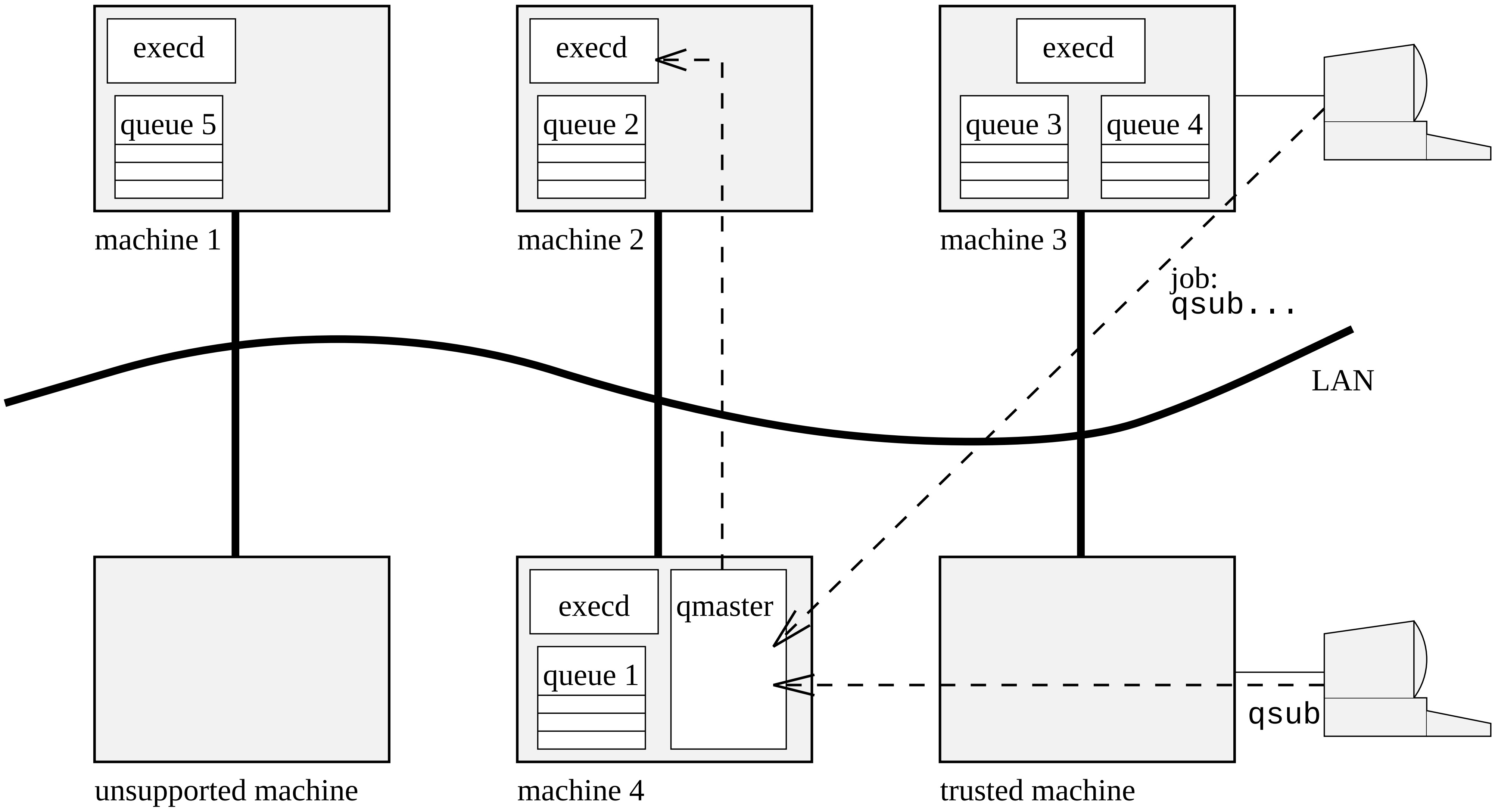
Institut für Informatik der Technischen Universität München
Lehrstuhl für Rechnertechnik und Rechnerorganisation
D-80290 München

Tel.: ++49-89/21052382, Fax: ++49-89-21058232
stellner@informatik.tu-muenchen.de

Queuing Systems

- Resource management
  - usually in computing centers
  - during batch processing
  - load leveling
- User interaction
  - based on the notion of *jobs*
  - specification of requirements → abstraction from the concrete execution platform
  - process migration based on checkpoint/restart facility
  - combining batch with interactive processing
- Existing systems
  - DQS
  - NQS
  - Condor
  - Codine

Codine


execd
queue 5
machine 1
execd
queue 2
machine 2
execd
queue 3
queue 4
machine 3
job:
qsub...
LAN
execd
queue 1
qmaster
qsub
unsupported machine
machine 4
trusted machine

Open Problems

⊖ Parallel jobs reduce interactive usability of workstations → users refrain from opening their machine

⊖ Checkpointing restricted to sequential programs → queuing systems cannot be used with parallel applications

Combination of Message-Passing and Queuing Systems

⊕ Separate process creation/management from communication

⊕ Checkpoints of parallel applications

- fault tolerance
- uninfluenced interactive work: migrating processes with starting interactive usage
- dynamic load balancing

⊕ Parallel jobs increase number of processes: even load distribution can be achieved easier

Consistent Checkpoints

Informal Definition

- Set of checkpoints of all processes of a parallel application
  - restart application correctly
  - no message lost or duplicated

Integrating Message-Passing and Queuing Systems

Managing Parallel Jobs

- Notion of *parallel job*: several processes form a management unit
- Submitting a parallel job
  - explicit: multiple processes upon submission are treated as a management unit
    - ⊖ only suitable for small numbers of processes
    - ⊖ dynamically created processes are not supported
    - ∗ low implementation effort
  - implicit: submit a single process and add dynamically created processes to the parallel job
    - ⊖ conflicts with message-passing libraries
    - ⊖ overhead due to queuing system interference
    - ⊕ flexible
    - ⊕ existing applications often use dynamic process creation during startup
- Scheduling
  - gang scheduling
  - new heuristics to respect needs of parallel applications
  - intra application load balancing

Creating Consistent Checkpoints

Assumptions and Properties

- First come first serve scheduling
- No failure during checkpoint creation → one checkpoint file
- Network file system (NFS, AFS, ...) used to transport checkpoint files
- Method suitable for
  - checkpointing (user initiated and periodic) and dynamic load balancing
  - great variety of message-passing libraries
- Transparent
- User level implementation (single process checkpointing from queuing system and overlayed functions)
- Strictly synchronizing approach

Overlayed Functions

Basic Idea

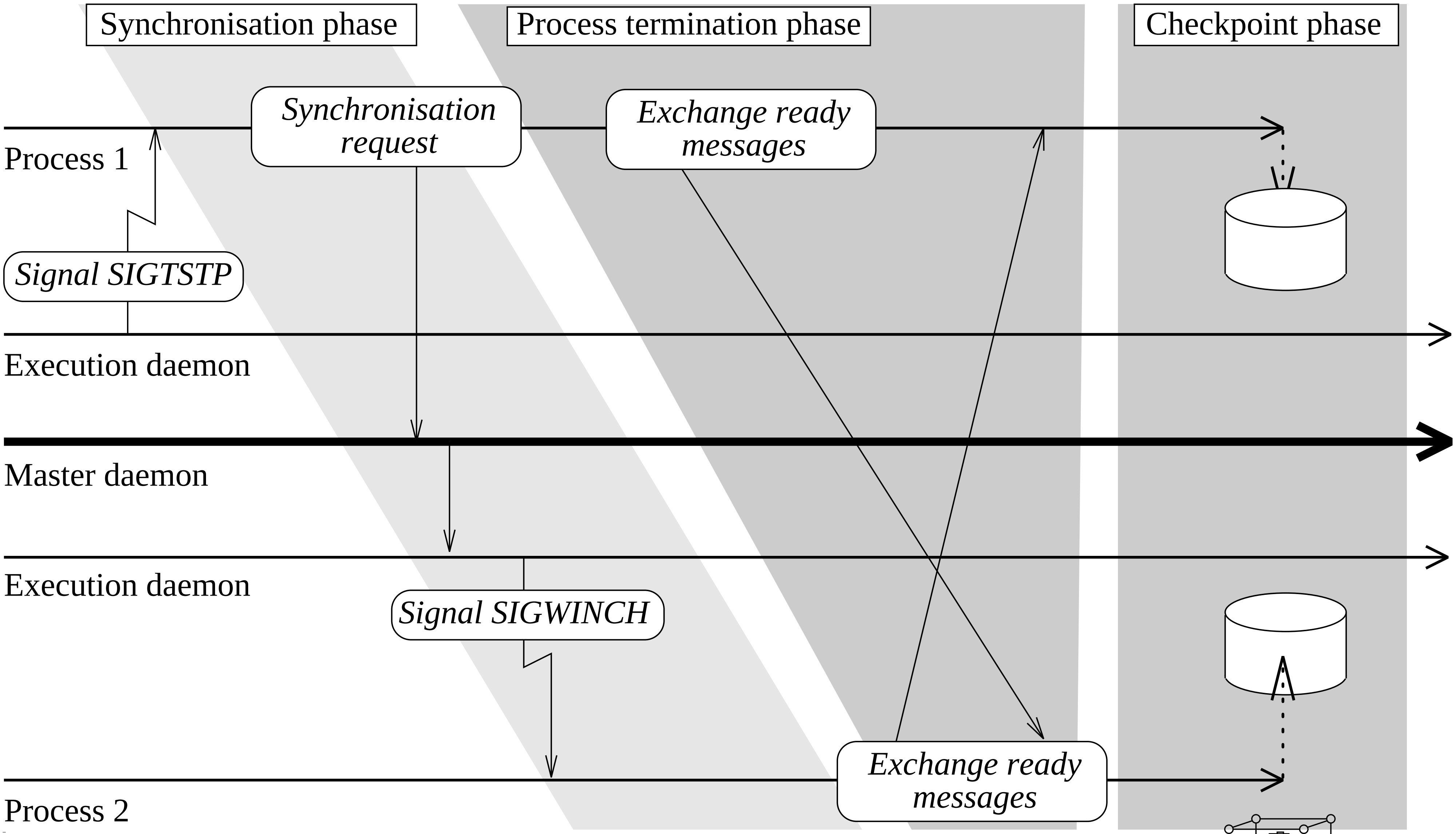
- Problems
  - frequently new versions of message-passing libraries are published
  - support different message-passing libraries
  - not all information in user address space

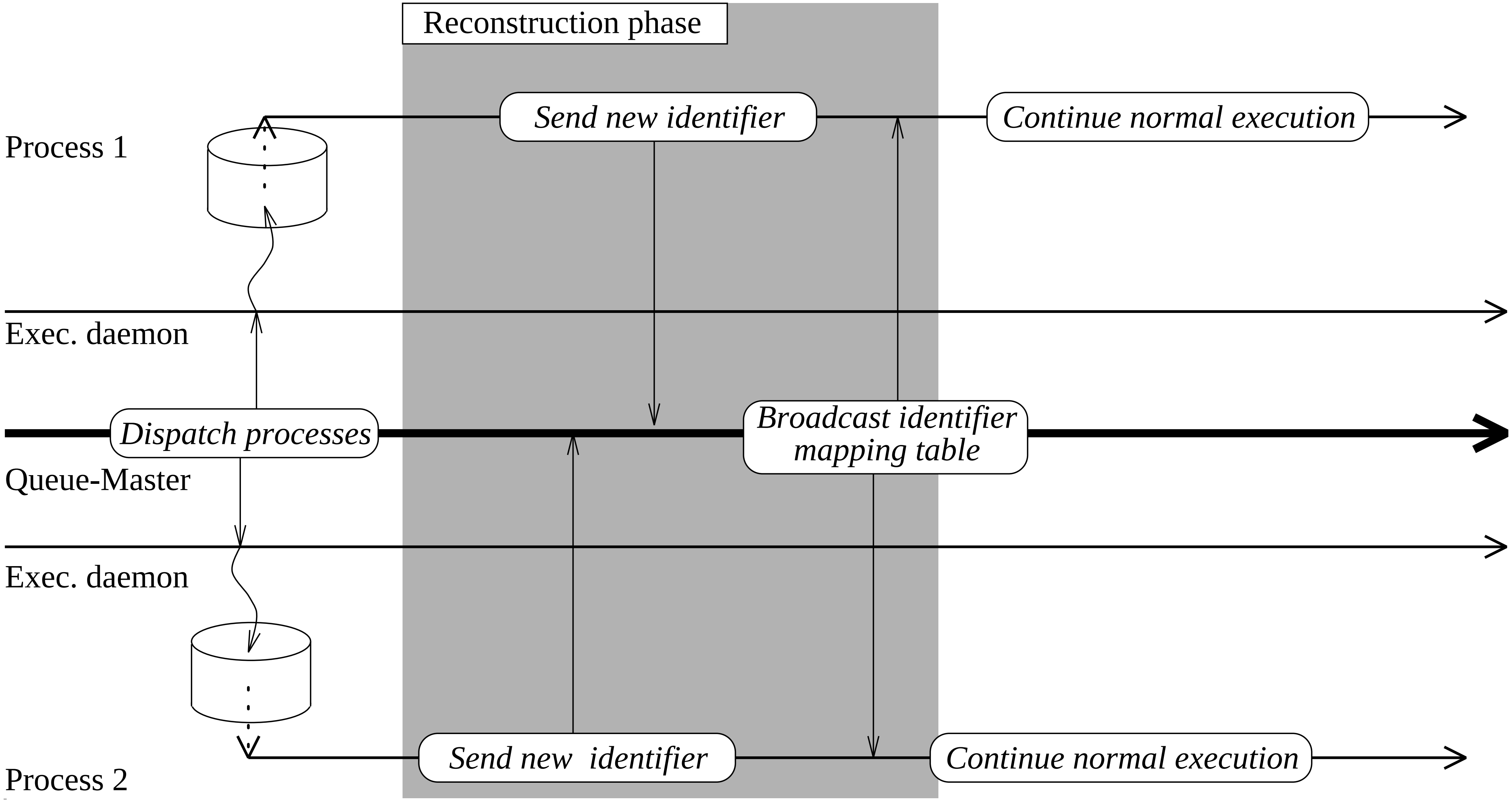
⇒ *Overlayed functions*
  - rename original library call
  - provide a stub doing additional work
    * logging parameters in user address space
    * modify parameters
    * call original library call by its new name
    * record results

Example

```
int syscall (a, b, c, d)
{
   int res;

   LogParameter (b, c); /* remember those parts in user address space */

   a = GetOrigValue (a); /* map to original descriptor */

   res = Syscall (a, b, c, d); /* invoke original function */

   LogResult (res, b, c); /* remember the result */

  return (res);
}
```

Implementation




Reconstruction phase
Process 1
Send new identifier
Continue normal execution
Exec. daemon
Dispatch processes
Broadcast identifier mapping table
Queue-Master
Exec. daemon
Send new identifier
Continue normal execution
Process 2

Prototype Evaluation

Environment and Results

- Research prototype for
  - UDP sockets
  - TCP sockets
  - PVM 3.1
- Measurements
  - DEC Alpha AXP 300 with OSF/1 version 1.2A
  - connected with Ethernet
  - PVM 3.1
  - Codine 1.3
  - communication intensive test application
- Results
  - 5 to 15 seconds to create a consistent checkpoint
  - 2 to 5 seconds to restart

Conclusion and Future Work

- Research prototype integrating PVM in Codine
  - protocol is viable → consistent checkpoints
  - starting interactive usage → processes moved to other machines
  - gain experience about users' requirements concerning process management

- Performance improvements

- Further message-passing libraries

- Dynamic load balancing

- Shift process creation/management to queuing systems