1 Introduction

2 Related work

3 ManyMan
   - Back-end
   - Front-end

4 Results
   - Migration
   - Energy

5 Conclusion
Requirements

1. Monitor the system:
   - Load, state and power consumption
   - Resource usage for each core
   - Resource usage per task
   - Overview of running, waiting, completed and failed tasks

2. Manage the system:
   - Create a task
   - Start a task on a specified core
   - Migrate tasks
   - Modify frequency and voltage
Monitor the system:

- Load, state and power consumption
- Resource usage for each core
- Resource usage per task
- Task output
- Overview of running, waiting, completed and failed tasks
Requirements

1. Monitor the system:
   - Load, state and power consumption
   - Resource usage for each core
   - Resource usage per task
   - Task output
   - Overview of running, waiting, completed and failed tasks

2. Manage the system:
   - Create a task
   - Start a task on a specified core
   - Migrate tasks
   - Modify frequency and voltage
Related work

- sccPerf from the sccKit
- Multi-core trace analysers
- Grid management systems

**Figure:** Intel’s sccPerf.
ManyMan
The Many-core Manager

- Modular:
  - Back-end (for SCC)
  - Front-end (for multi-touch)
  - Communication layer (JSON)
- Easy creation of other front- and back-ends
ManyMan

Back-end

- Written in Python
ManyMan
Back-end

- Written in Python
- SSH for task creation on the SCC from MCPC
  - New ssh process for each task
ManyMan
Back-end

- Written in Python
- SSH for task creation on the SCC from MCPC
  - New ssh process for each task
  - SSH connection sharing improves latency
ManyMan
Back-end

- Written in Python
- SSH for task creation on the SCC from MCPC
  - New ssh process for each task
  - SSH connection sharing improves latency
- Monitoring
  - top
  - scCBmc
ManyMan
Back-end

- Written in Python
- SSH for task creation on the SCC from MCPC
  - New ssh process for each task
  - SSH connection sharing improves latency
- Monitoring
  - top
  - sccBmc
- Managing
  - Berkeley Lab Checkpoint/Restart (BLCR)
  - Custom tool for frequency scaling
ManyMan - Front-end
Chip Overview

Intel SCC
48-core chip

Add task
Pending tasks:

- Count New
- Sleepy greeter New
- Memory New
- Pi New

Help
Exit

ManyMan
Many-core Manager

Finished tasks:

- Pi
  Status: Finished

- Pi
  Status: Finished

Set frequency
Overall Power-usage:
Power: 31W

Overall CPU-usage:
CPU: 5%
ManyMan - Front-end
Core Details

Intel SCC
48-core chip

Add task

Pending tasks:
- Count New
- PI New
- Memory New
- Memory New

Overall CPU-usage:
CPU: 4%

Core 28
CPU: 100%
MEM: 15%

Finished tasks:
- Memory Status: Finished
- PI Status: Finished
- PI Status: Finished
- PI Status: Finished

Set frequency

Overall Power-usage:
Power: 18W
ManyMan - Front-end

Core Details

Core 28

CPU: 100%

MEM: 15%

Pi
Status: Running

Memory
Status: Running

Sleepy greeter
Status: Running

Pi
Status: Stopped
ManyMan - Front-end

Task Creation

Add task

Name (optional): Test task

Command: /path/to/executable -flag value
ManyMan - Front-end
Frequency Scaling

Intel SCC
48-core chip

Pending tasks:
- Count: New
- Sleepy greater: New
- Memory: New
- Pi: New

Power domain 0:
Frequency: 200MHz

Power domain 1:
Frequency: 100MHz

Power domain 2:
Frequency: 800MHz

Power domain 3:
Frequency: 320MHz

Power domain 4:
Frequency: 267MHz

Power domain 5:
Frequency: 400MHz

All power domains:
Frequency: 533MHz

Overall CPU-usage:

CPU: 0%

Completed tasks:
- Memory:
  Status: Finished
- Pi:
  Status: Finished

Overall Power-usage:
Power: 26W
ManyMan - Front-end

Frequency Scaling

![Image of frequency scaling settings]

- Power domain 0: Frequency: 200MHz
- Power domain 1: Frequency: 100MHz
- Power domain 2: Frequency: 800MHz
- Power domain 3: Frequency: 320MHz
- Power domain 4: Frequency: 267MHz
- Power domain 5: Frequency: 400MHz
- All power domains: Frequency: 533MHz
Results

Migration using BLCR

![migration_graph]

- Checkpoint NFS
- Restart NFS

![migration_graph2]

- Checkpoint NFS
- Checkpoint /tmp
- Restart NFS
- Restart /tmp
Results

Power Consumption

- Voltage (V) vs Frequency (MHz)
- Power (W) vs Frequency (MHz)
- FLOPs vs Frequency (MHz)
- MFLOPs/W vs Frequency (MHz)
Conclusion

- No suitable visualization and management tools around
- BLCR allows for easy migration, but migrating is very expensive
- Connection Sharing speeds up connecting
- Easy voltage and frequency scaling, maximum efficiency at 400 MHz
- Created an intuitive tool for research or education
Questions?

Demo video and source download available:
http://www.science.uva.nl/~jimivdw/manyman/