

OS and I/O for Merrimac

Mendel Rosenblum

Stanford University

What does an OS do?

- Export nice abstractions to the programmer/user.
- Manage the hardware resources (memory, processors, etc.)
 - Memory placement issues are important.
- Handle input/output.

What is Merrimac?

- Computer system or an accelerator/attached processor?
 - Does it run all the computation or just pieces of it?
 - Is there a host environment we can leverage?

What abstractions are export?

- What APIs need to be support?
 - Brook runtime environment?
 - Linux runtime environment?
 - MPI or UPC?
- What kind of legacy language/code support do we need?
- How much does OS do automatically vs. simple expose to the programmer?

What can we get away with?

- Can probably get away with exposing more of the raw hardware.
- Brook runtime environment?
- Posix environment?
- Legacy support?

Hardware resource management

- Single task versus multi-tasking?
 - Is there machine going to be dedicated to a particular task for long periods of time?
- Do we want multiple task to share the machine?
 - Time sharing or space sharing?
- How does this effect the programming model (static vs. dynamic assignment)

Placement support

- Do we need a level of indirection from programmer visible names from nodes and memory to real nodes and memory?
- Example: UPC data placement.

Input/Output

- Can we get away with “only I/O device is network?”
- Do we need to stream from I/O devices?
 - Is main memory like a second-level SRF?
 - Is there some bandwidth vs. computation ratio augment here?

Input/Output (2)

- How does I/O work with the data placement?
- Do we need to convert data from storage format to main format and layout?

Construction techniques

- Depends on our needs.
- Choices
 - Port Linux kernel to machine
 - Get Linux runtime environment
 - Roll your own loader/runtime
 - Port Brook runtime to machine

Discussion

- Given the risk already in Merrimac (e.g. new language, new architecture), fancy new OS technology might be too much.
- Exposing the machine resources with a thin layer seems like a good idea.
- Need to figure out I/O requirements.