1

Managing the Processors

1.1

Having an operating system in this HPC environment is absolutely necessary. With such a large number of nodes and processors available, requires a managed resource system that an operating system provides, not to mention the added benefits of allowing multiple programs to interact with a given node, or scheduling of programs for individual nodes or processors as needed. The programming overhead for managing these nodes individually, especially considering the number of nodes and potential for cross-program interference, is a nightmare. Stated simply, regardless of which precautions you may take in your own program, without an operating system, a malicious user, such as a disgruntled programmer unable to design their own operating system, may decide to alter your program's state and cause any program you run to fail in a variety of maddening and awful ways.

Even assuming that hardware failures do not occur in such a massive system, and that no other users get any time on the system while you have that time, removing the abstraction of hardware access would require far more programming time than would be gained from removing that lowest level of abstraction.

Space-time (Non?)continuum

1.2

Time-shared in this context is context switching among

Space-shared

1.3

Time-sharing on the HPCOS would give the advantage of allowing multiple processes to run on

1.4

Space-sharing on the HPCOS constitute

1.5

Different tasks running on a single compute node should be processes, to prevent interference in each others' memory space, as well as sustain the principle of least privilege. Using threads would require that all processes use the same memory space, part of the reason that an Operating System exists in the first place, . In the case of some HPCOSs, only one task is given to a compute node or processor,

in which case that process should still be a process for the purpose of preventing a possible system failure. In that event, if the Operating System were running the process as a thread, recovery would be considerably more difficult, and the entire compute node might need to be restarted.

1.6

Tasks running across different nodes are required to be processes. Any communication between these processes would have to be explicit communication.

Managing Memory

4.1

4.2

4.3

4.4

4.5

4.6

Managing Storage

5.1

A RAID 0 scheme in this context would work by distributing data among the external network servers, and striping them as though each fileserver was an individual drive. This scheme would be beneficial in preventing bandwidth loss, though it would cause a huge number of problems in case any system failed.

A wise choice in this plan would be to ask that the NFS as a whole use a RAID 1 setup, for the most reliable, or a RAID 5 setup, for improved general performance, as well as some drive failure resistance.

5.2

5.3

5.4