Subject: Re: [PATCH 1/2] rcfs core patch
Posted by Paul Jackson on Fri, 09 Mar 2007 02:35:07 GMT
View Forum Message <> Reply to Message

Herbert wrote:
> why is the filesystem approach so favored for this
> kind of manipulations?

I don't have any clear sense of whether the additional uses of file systems being considered here are a good idea or not, but the use of a file system for cpusets has turned out quite well, in my (vain and biased ;) view.

Cpusets are subsets of the CPUs and memory nodes on a system.

These subsets naturally form a partial ordering, where one cpuset is below another if its CPUs and nodes are a subset of the other ones.

This forms a natural hierarchical space. It is quite convenient to be able to add names and file system like attributes, so that one can do things like -name- the set of CPUs to which you are attaching a job, as in "this job is to run on the CPUs in cpuset /foo/bar", and to further have file system like permissions on these subsets, to control who can access or modify them.

For such hierarchical data structures, especially ones where names and permissions are useful, file systems are a more natural interface than traditional system call usage patterns.

The key, in my view, is the 'shape' of the data. If the data schema is basically a single table, with uniform rows having a few fields each, where each field is a simple integer or string (not a fancy formatted string encoding some more elaborate shape) then classic system call patterns work well. If the schema is tree shaped, and especially if the usual file system attributes such as a hierarchical name space and permissions are useful, then a file system based API is probably best.

--
I won't rest till it's the best ...
Programmer, Linux Scalability
Paul Jackson <pj@sgi.com> 1.925.600.0401