Programmable File System Workshop HPDC ‘14

Fishbowl discussion panel

Part of the program for the Programmable File System Workshop at HPDC ‘14 was a fishbowl panel session where we discussed a variety of topics. What follows are my bullet points that summarize the topics of conversation.
Brent Welch

What are the high level goals for Programmable FS?

Data management for file systems
Data model transcendent systems (a la database communities approach)
Shipping code into the file system (non-closed data model, hard to optimize)
Imposing a more sophisticated data model (i.e., non-POSIX)
Need to model knowledge of the data (i.e., what workflows are done with it)
Can the system generate the descriptive metadata for users?
4000-6000 data formats across the corpus of government documents on their websites

File System vs Storage System (Database, Key-Value, Block Storage, File Storage, Image Storage, …)

Each week there is a new software defined storage company that will change the world
Different kinds of storage for different kinds of data, especially metadata that has structure

Blessing and Curse of a stable API (POSIX)

POSIX is a fine interface for bits/bytes
Huge number of Web based (REST) API
Durability of the HTTP GET/POST/LIST URL framework
Commodity block API forces innovation above that layer
New devices - Seagate with its Ethernet interface / REST / static metadata (objects, no containers, key-value, peer-to-peer support)
HGST - a micro server (web server, can run user code on Linux 3.2)
New devices have lead to new work in local file systems
Going around the internal Linux VFS interface
Device interface vs. Service interface

SQL lets “users” program data accesses - what is the analogy in File Systems ?

Declarative SQL model hides complexity, allows for automatic optimization
I/O Libraries (netCDF) provide models that are aligned with their users - feeds a schema to the FS

Extensible systems have “hooks” and callback points where application code can be inserted into infrastructure code - (e.g., DB stored procedure) - what are motivating applications for File Systems?

72 policy hook points in iRODS (11 or 12 commonly used)

Filtering? Multi-level cache control? Load sharing and sharding?

Fat Stack

Speed of SSD and networking has highlighted the thickness of the storage stack
Solutions like RAMcloud provide memory based key-value that is persistent to disk
“Better never than late” - “The tail at scale”
Discussion of long tail-latency in big systems

Metadata

Horizon system in HPDC 2010, deadline information flowing through the system
What kind of metadata should flow through the system with the data?

Ideas for future workshops

What is the API between the Database and the File System (e.g., double-write problem)
Folks are writing big systems with good performance - but what users are stuck “out in the cold” - what are the motivating use cases that drive FS improvements
Creative use of POSIX - e.g., PLFS
API - impact on performance, impact on usability
Overall integration of storage systems and their management systems - can we optimize this overall, very complex stack
Get more application developer participation - Domain Specific Languages
Optimizing HPC applications without actually changing them - e.g., “skinning” netCDF
Exascale 10 - library approach to building new storage systems github.com/exascale10/design
FastForward - Intel has a wiki with design documents - three independent groups (HDFS5, DOAS, IOD)
Building block approach for hard things like transactions, journals, recovery (similar to BOOST library in spirit)

References during Discussion (Thanks Nathan Rutman!)

The Tail at Scale
http://research.google.com/pubs/pub40801.html
https://www.youtube.com/watch?v=C_PxVdQmfpk

A Study of Linux FS Evolution
http://pages.cs.wisc.edu/~ll/papers/fsstudy.pdf