Programmable Storage

Carlos Maltzahn, UC Santa Cruz

10/6/2020
Snowmass Community Planning Meeting

Opportunities for computing R&D to advance particle physics
Carlos Maltzahn

people.ucsc.edu/carlosm

Introduction

Dr. Carlos Maltzahn is the founder and director of the UC Santa Cruz Center for Research in Open Source Software (CROSS). Dr. Maltzahn also co-founded the Systems Research Lab, known for its cutting-edge work on programmable storage systems, big data storage & processing, scalable data management, distributed system performance management, and practical reproducible evaluation of computer systems. Carlos joined UC Santa Cruz in 2004, after five years at Netapp working on network intermediaries and storage systems. In 2005 he co-founded and became a key mentor on Sage Weil's Ceph project. In 2008 Carlos became a member of the computer science faculty at UC Santa Cruz and has graduated nine Ph.D. students since. Carlos graduated with a M.S. and Ph.D. in Computer Science from University of Colorado at Boulder.

His work is funded by government and industry, including NSF OAC-1836659, NSF CNS-1705621, DOE ASCR DE-0001705910, NSF OAC-1450488, and CROSS.

For more details, you can read his vitae.

Administrative Staff
- Stephanie Lucig
- Lavinia Preston

Research Staff
- Kate Compton
- Ivo Jimenez

Current Ph.D. Students
- Saided Adelujio
- Xiaowei Chu
- Jianshen Liu
- Emaaef Darling
- Yiming Chang

Graduated Ph.D. Students
- Alexandre Arons, 2011 (thesis)
- Joe Buck, 2014 (thesis)
- Adam Crums, 2015 (thesis)
- Latches Aronov, 2018 (thesis)
- Ivo Jimnez, 2019 (thesis)
- Michael Sevilla, 2018 (thesis)
- Andrew Shevman, 2016 (thesis)
- Dimitrios Skoutis, 2014 (thesis)
- Noah M. Watkins, 2018 (thesis)
Computational Storage: History

- Idea dates back to mainframes
  - First Channel I/O processors in IBM 709, 1957
- Network Attached Secure Disks (NASD)
  - Research project at CMU, 1997-2001
    - Encryption, compression, data management (“active storage”)
  - SCSI T10 Object Storage Device (OSD) v1 and v2 standards
    - Only offloads part of file system functionality
- Ceph
  - Research project at UC Santa Cruz, 2005-2007
  - Designed for OSDs
  - Broke OSD standard with P2P communication for failure management
  - Implemented for hosts, not devices
- SkyhookDM Plugin for Ceph
  - CROSS incubator project at UC Santa Cruz since 2016
  - Offloads data management of tabular data
  - Turns Ceph into an Apache Arrow-native store (since 2020)
- Computational Storage
  - SNIA Technical Working Group (TWG) since 2019
  - Focus on storage devices
- Eusocial Storage Devices
  - CROSS research project at UC Santa Cruz since 2017
  - P2P communication, specialization into “castes”
  - I/O stack flexible about offloading: pushdown, pushback
  - Leverages Smart NICs
Computational Storage: Why now?

- Storage devices are getting very fast
  - CPU/DRAM/PCIe cannot keep up
  - CPU/DRAM/PCIe tax for storage increases

- Disaggregation in data centers
  - Multi-tenant workloads are too diverse for any kind of packaging
  - Better to dynamically assemble systems from parts

- Storage fabrics are expensive
  - NVMe requires host kernel resources
  - Ethernet is much cheaper and keeps getting faster
  - New IP protocols are getting very fast: e.g. HTTP/3
Programmable Storage

A programmable storage system or device exposes internal subsystem abstractions as “interfaces” to enable the creation of higher-level services via composition.

Collaborators: Jeff LeFevre (UCSC), Ivo Jimenez (UCSC), Esmaeil Mirvakili (UCSC), Jayjeet Chakraborty (NIT), Aditi Gupta (NIT), Aaron Chu (UCSC), Xiongfeng Song (Rice)
Programmable Storage

Computational Storage + Programmability

For storage systems:
- Storage has to be correct, otherwise data loss
- Correct software takes time
- Reuse as much as possible → Composability
- Composability important for optimization

For storage devices:
- Storage device industry has very low margins
- Products must fit existing market and have a minimum life time
- Programmable devices to reduce market risk
- Greater opportunity for innovation
What is SkyhookDM?

An object “class” for Ceph
- No upstream modifications required
- Inherits Ceph’s properties now and in the future
- Can use all other object extensions
- **Not a database**

Storing *tabular* data in objects
- Using **ARROW**

Object read/write operations
- Select, Project, Aggregate
- Create, append rows/columns
- Indexing
- Intra- & inter-object transformations

Growth of mainline object classes 2010-2018
SkyhookDM Client

- Client maps *tables* to *sets of objects*
  - Map is also stored in objects

- Client API designed for *plugins*
  - Allows *pushdown* to *scale out* tabular data operations
  - Reduces data movement (CPU cycles!)

- IRIS-HEP
  - Connecting to Coffea and ServiceX

- CROSS
  - Plugins for Postgres, Spark, Pandas, HDF5
How does SkyhookDM fit into DOMA?

ServiceX Plugin and Coffea Processor:
ServiceX creates one table per *transformation request*
- Partitions table and assigns transformer to each partition
- Each transformer creates and writes an object row-by-row
- ServiceX provides table metadata, incl. partitioning to SkyhookDM

Table names are arbitrary strings, globally unique
- Column names are arbitrary strings, unique within table
- Rows have a key, unique within table

Coffea Processor:
SkyhookDM can create views across arbitrary sets of tables
- View names are arbitrary strings, globally unique
- Views can be either by reference or by copy (i.e. materialized)
- SkyhookDM stitches views on a best-effort basis
- Key mappers can map one kind of key to another and can be stored

Design allows evolution of higher-level automation.
- Table naming conventions might indicate compatibility to other tables
- View naming conventions might allow automatic reuse of materialized views
- Column naming conventions might allow versioning
- Naming convention might allow automatic garbage collection