DeltaFS Indexed Massive Dir

Software-Defined Storage For Fast Query

PDSW-DISCs 2017

Qing Zheng, George Amvrosiadis, Saurabh Kadekodi, Michael Kuchnik
Chuck Cranor, Garth Gibson
Brad Settlemyer, Gary Grider, Fan Guo

Carnegie Mellon University
Los Alamos National Laboratory (LANL)
DeltaFS Indexed Massive Dir

Key features

1. Require no dedicated resources
2. Almost no post-processing is needed
3. Low I/O overhead

http://www.pdl.cmu.edu/
DeltaFS Indexed Massive Dir

Target workloads

1. Data-intensive HPC simulations
2. Not designed for indexing checkpoints
3. I/O bandwidth is limited

http://www.pdl.cmu.edu/
Agenda

Part 1 – Motivation
Part 2 – In-situ indexing design
Part 3 – API, LANL VPIC integration
Conclusion

http://www.pdl.cmu.edu/
Existing HPC builds indexes during post-processing

Delay queries until post-processing done (5-20% simulation time)

http://www.pdl.cmu.edu/
Problem faced:
The increasing time-to-science

Due to the growing gap between compute and I/O
Inefficient support on small data

simulation start  query finish
Processing data in-transit while data is written to storage

Need separate resources for sorting and indexing

- http://www.pdl.cmu.edu/
In-situ indexing directly on app nodes using app resources

No need for a separate indexing cluster

http://www.pdl.cmu.edu/
Key idea:
Reuse storage write-back buffering and idle CPU cycles for in-situ indexing
Example app: LANL VPIC


Particle: 40 bytes

PDSW-DISCS 2017

http://www.pdl.cmu.edu/
TBs I/O per trajectory fetch

file-per-process

Simulation procs

One output file per VPIC process

Data object

Query a single particle trajectory

TBs search

http://www.pdl.cmu.edu/
5,000x faster than baseline with DeltaFS in-situ indexing

Time for reading a single particle trajectory (10TB, 48 billion particles)
Part II

System design: Light-weight in-situ indexing

1. Tiny mem footprint
2. Zero write amplification
3. No read back
Resource-efficient indexing by log-structured I/O

Tiny mem footprint, full storage b/w util.

http://www.pdl.cmu.edu/
LSM-Trees compacts all the time, but we can’t afford it. Must aim for low I/O overhead at 10%-20%.

Compaction easily causes 1000% I/O overhead by reading/writing previously written data.

http://www.pdl.cmu.edu/
In-situ indexing by aggressive data partitioning

All-to-all shuffle

Bound the number of data needed per query per timestep

App process #0

App process #1

App process #2

Compute I/O Compute I/O

Bound the number of data needed per query per timestep

http://www.pdl.cmu.edu/
In-situ indexing as a file system lib component

No dedicated cluster needed

App data
shuffle sender

All-to-all shuffle
shuffle receiver

WriteBuffer

Index Log
Data Log

index block
filter
data block
data block

http://www.pdl.cmu.edu/
Part III

Programming interface:
Indexed Massive Directory (IMD)

In-situ indexing keyed on filenames

mkdir("./particles", DELTAFS_IMD)
How to use Indexed Massive Dir (IMD)

1. Data searched together go into a single IMD file
   e.g. one file for each particle

2. Create as many IMD files as you want
   e.g. 1 trillion files for 1 trillions particles

Query your data by “open-read-close”
VPIC using DeltaFS IMD

**file-per-particle**

Simulation procs

1M

One IMD file per VPIC particle

Indexed Massive Directory

Index object
data object

Index object

Data object

http://www.pdl.cmu.edu/
LANL Trinity Experiments

- VPIC-Baseline
  - VPIC
  - buffer
  - DeltaFS indexing

- VPIC-DeltaFS
  - VPIC
  - buffer

- Compute Node
  - 32 cores/node

- Burst-buffer
- SSD
- HDD
- Lustre

- No post-processing

- Queries

- 1-99 compute nodes, 496 million - 48 billion particles

- http://www.pdl.cmu.edu/
Baseline (Full-system parallel scan)

DeltaFS (w/ 1 CPU core)

Query Time (sec)

Simulation Size (million particles)

- 1 node
- 2 nodes
- 4 nodes
- 8 nodes
- 16 nodes
- 33 nodes
- 66 nodes
- 99 nodes

1 node: 496, 992, 1,984, 3,968, 7,936, 16,368, 32,736, 49,104

http://www.pdl.cmu.edu/
I/O Time per Dump (sec)

- **Baseline**
- **DeltaFS**

**Tiny simulations**
- 9.63x
- 4.78x
- 2.42x
- 1.56x

**Bigger simulations**
- 1.29x
- 1.13x
- 1.15x
- 1.13x

Simulation Size (million particles)
- 1 node
- 2 nodes
- 4 node
- 8 node
- 16 nodes
- 33 nodes
- 66 nodes
- 99 nodes

Simulation Sizes:
- 496
- 992
- 1,984
- 3,968
- 7,936
- 16,368
- 32,736
- 49,104

[http://www.pdl.cmu.edu/]
Conclusion

In-situ indexing for transparent, almost-free query acceleration
  no dedicated nodes, no post-processing, ~15% I/O overhead

• Indexed Massive Dir (~3% app mem, compaction-free, POSIX API)
• Powered by Mercury RPC
  https://mercury-hpc.github.io/
• DeltaFS is one of the Mochi micro-services

https://github.com/pdlfs/deltafs

http://www.pdl.cmu.edu/