

# Analyzing I/O Performance on a NEXTGenIO Class System

holger.brunst@tu-dresden.de ZIH, Technische Universität Dresden

LUG17, Indiana University, June 2<sup>nd</sup> 2017

## **NEXTGenIO Fact Sheet**



## Project

- Research & Innovation
  Action
- 36 month duration
- €8.1 million

### **Partners**

- EPCC
- INTEL
- FUJITSU
- BSC
- TUD
- ALLINEA
- ECMWF
- ARCTUR



## Approx. 50% Committed to Hardware Development



- n|e|x|t|g|e|n|i|o
- Intel<sup>™</sup> DIMMs are a Key feature
  - Non-volatile RAM
  - Much larger capacity than DRAM
  - Slower than DRAM
    - By a certain factor
    - Significantly faster than SSDs ™
  - 12 DIMM slots per socket
    - Combination of DDR4 and Intel<sup>™</sup> DIMMs
- How can files systems like Lustre benefit?

## **Three Usage Models**



- The "memory" usage model
  - Extension of the main memory
  - Data is volatile like normal main memory
- The "storage" usage model
  - Classic persistent block device
  - Like a very fast SSD
- The "application direct" usage model
  - Maps *persistent* storage into address space
  - Direct CPU load/store instructions

## New Members in Memory Hierarchy



- New memory technology
- Changes the memory hierarchy we have
- Impact on applications e.g. simulations?
- I/O performance is one of the critical components for scaling application
   Lustre serves as primary

file system



## **Using Distributed Storage**



- Lustre global file system
  - No changes to apps
  - Support for NVRAM?
- Required functionality
  - Create and tear down file systems for jobs
  - Works across nodes
  - Preload and postmove filesystems
  - Support multipl across system
- I/O Performance
  - Sum of many la,



## Using an Object Store



Memory

Node

Memory

Node

- Needs changes in apps
  - Needs same functionality as global filesystem
  - Removes need for POSIX functionality
- I/O Performance
  - Different type of abstraction
  - Mapping to objects
  - Different kind c Instrumentatio



Memory

Memory

Node

Memory

Node

Memory

Node

## **Towards Workflows**



- Resident data sets
  - Sharing preloaded data across a range of jobs
  - Data analytic workflows
  - How to control access/authorisation/securi ty/etc....?
- Workflows
  - Producer-consumer model
  - Remove file system from intermediate stages
- I/O Performance
  - Data merging/integration?





- Analysis tools need to
  - Reveal performance interdependencies in I/O and memory hierarchy
  - Support workflow visualization
  - Exploit NVRAM to store data themselves
  - (Workload modelling)





## Vampir & Score-P





June 2nd, 2017

## Tapping I/O Layers

- I/O layers
  - Lustre File System
    - Client side
    - Server side
  - Kernel
  - POSIX
  - MPI-I/O
  - HDF5
  - NetCDF
  - PNetCDF



## Lustre I/O performance data



## • Event data AND aggregated numbers

- Open/Create/Close operations (meta data)
- Data transfer operations
- Client side
  - procfs via PAPI-Components counters, Score-P (user space)
- Server side
  - Data base (custom made, system space)
  - Mapping to nodes/applications

## What the NVM Library Tells Us



- Allocation and free events
- Information
  - Memory size (requested, usable)
  - High Water Mark metric
  - Size and number of elements in memory
- NVRAM health status
  - Not measurable at high frequencies
- Individual NVRAM load/stores
  - Remain out of scope (e.g. memory mapped files)

## I/O Operations over Time







## I/O Data Rate over Time









## I/O Summaries per File



#### All Processes, Aggregated I/O Transaction Time per File Name 3.0 s 1.5 s 0s 3.628 s 2.503 s 2.027 s 1.986 s 1.359 s 1.182 s 0.942 s 0.798 s 0.768 s 0.704 s 0.694 s 0.59 s

/N/dc2/scratch/wardmod/input/fineGrid/frate.wheat.nc /N/dc2/scratch/wardmod/input/fineGrid/frate.corn.nc /N/dc2/scratch/wardmod/input/fineGrid/frate.soy.nc /N/dc2/scratch/wardmod/inp...dx/daily/Intr relh 1948.nc /N/dc2/scratch/wardmod/inp...dx/daily/Intr\_rads\_1948.nc /N/dc2/scratch/wardmod/in...x/daily/Intr\_wspd\_1948.nc /N/dc2/scratch/wardmod/i...C 60year climo mm day.nc /tmp/mpi/proc6/output/hourly/1948\_hourly4thmb.nc /N/dc2/scratch/wardmod/inp...dx/daily/Intr\_tmin\_1948.nc /N/dc2/scratch/wardmod/inp...dx/daily/Intr\_tmax\_1948.nc /N/dc2/scratch/wardmod/i...ntr RADS 60year climo.nc /N/dc2/scratch/wardmod/i...ntr WSPD 60year climo.nc /N/dc2/scr ch/wardmod/i...ntr TMAX 60year climo.nc

Aggregated data for specific resource

4.5 s

5.232 s

LUG17

## I/O Operations per File







## Taken from My Daily Work...



- Bringing the Lustre system I/O down
  - with a single (serial) application
- Higher I/O demand than IOR benchmark
- Why?



## Coarse Grained Time Series Reveal Some Clue, but...





## **Details Make a Difference**





Master thread

# Approaching the Real Cause

80 KiB

137.812 KiB

136 KiB /N/dc2/scratch/wardmod/input/fineGrid/zedx/daily/Intr\_prec\_1948.nc MAIN 1 1.812 KiB /N/dc2/scratch/wardmod/input/fineGrid/zedx/daily/Intr prec 1948.nc open incliget v 2 з Master thread 2 close. 3 ead fopen ...15! POSIX read operations June 2nd, 2017 22

0B

Sum





## Before and After...





June 2nd, 2017

## Summary



- NEXTGenIO developing a full hardware and software solution
  - Solution includes Lustre
- Performance focus
  - Consider complete I/O stack
  - Incorporate new I/O paradigms
  - Study implications of NVRAM
- Reduce I/O costs
- New usage models for HPC and HPDA



## **Supplementary Slides**

1.11 1.00

## NVRAM allocation over time





## MAP Timeline



Profiled: mmult2 sol c.exe on 8 processes, 1 node, 8 cores (1 per process) Sampled from: Wed Nov 9 2016 14:47:59 (UTC) for 16.1s



14:47:59-14:48:15 (16.120s): Main thread compute 42.4 %, MPI 56.2 %, File I/O 1.3 %

## Vampir Timeline





## Future Stats



II Processes	s, Number of Hits	per Source Code Locat	ion	
60 207 (14	IOOK	50 K		
160,307 (14.6%)				
134,400 (12.24%)				
	124,823 (11.379	6)	rhs.f:307	
	101,387	7 (9.23%)	rhs.f:328	
		81,371 (7.41%)	jacu.f:18	
		71,172 (6.48%)	jacld.f:32	
		51,809 (4.72	%) Unknowr	
		36,002 (3.28%)	jacld.f:10	
		31,540 (2.87%)	jacld.f:33	
		23,288 (2.12%)	rhs.f:318	
		20,820 (1.9%)	jacu.f:10	
		17,884 (1.63%)	jacld.f:18	
		16,526 (1.51%)	jacld.f:32	
		14,934 (1.36%)	rhs.f:312	
		14,159 (1.29%)	jacld.f:44	
		13,490 (1.23%)	iacld.f:27	
		11.814 (1.08%	) iacu f·18	
		11 233 (1 02%	) jacu f:33	

All Processes, Number of Hits per Source Code Location

3 k	2 k	1 k	0
3,173 (14.8	1%)		buts.f:238
2,7	/07 (12.63%)		blts.f:59
	2,411 (11.25%)		rhs.f:307
	1,884 (	8.79%)	rhs.f:328
	1,	611 (7.52%)	jacu.f:181
		1,417 (6.61%)	jacld.f:329
		990 (4.62%)	Unknown
		663 (3.09%)	jacld.f:105
		643 (3	%) jacld.f:335
		411 (1.92%)	rhs.f:318
		396 (1.85%)	jacu.f:105
		351 (1.64%)	jacld.f:181
		324 (1.51%)	jacld.f:328
		302 (1.41%)	jacld.f:278
		281 (1.31%)	rhs.f:312
		267 (1.25%)	jacld.f:44
		254 (1.19%)	jacu.f:187
		230 (1.07%)	iacu.f:331

## Visualization of memory access statistics



- Currently, Vampir has many different views presenting counter metrics
- Future: Absolute number of memory accesses performed by an application for diff types of memory





## **Future Stats**



hread, Values of Metric "long int\* vector" over Time



June 2nd, 2017

LUG17