

Michael Kluge, ZIH

# Correlating Multiple TB of Performance Data to User Jobs

#### Lustre User Group 2015, Denver, Colorado

Zellescher Weg 12 Willers-Bau A 208 Tel. +49 351 - 463 - 34217

Michael Kluge (michael.kluge@tu-dresden.de)



- ZIH: about us
  - -HPC and service provider
  - -Research Institute
  - Big Data Competence Center
- Main research areas:
  - -Performance Analysis Tools
  - -Energy Efficiency
  - -Computer Architecture
  - -Standardization Efforts (OpenACC, OpenMP)







2

#### ZIH HPC and BigData Infrastructure



#### HRSK-II, Phase 2, currently installed



#### Architecture of our storage concept (FASS)







# Performance Analysis for I/O Activities (1)

- What is of interest
  - Application requests, Interface types
  - Access sizes/patterns
  - Network/Server/Storage utilization
- Level of detail:
  - Record everything
  - Record data every 5 to 10 seconds
- Challenge:
  - How to analyze this?
  - How to deal with anonymous data?





## I/O and Job Correlation: Architecture/Software





https://fusionforge.zih.tu-dresden.de/projects/dataheap/



Michael Kluge, ZIH

## Visualization of Live Data for Phase 1





Michael Kluge, ZIH

Center for Information Services & High Performance Computing

## Visualization of Live Data for Phase 2

Phase 1 InfiniBand	Phase 2 InfiniBand	P1LUNs bandwidth P1 I dsz 1581 dsz	UNS loops P2 LUNS bandwidth	P2 LUNs bandwidth P2 LU	Ills bandwidth P2 LUIis bandwid 0 6414 0	th P2 LUNs is ops P2 de2 0 de8	2 LUNs icops P2 LUNs icops P2 LUNs icops P2 SSD bandwidth P2 SSD icops rats 0 rats 0 rats 0	-100%
IB Empfang in MB/s 6130	IB Empfang in GB/s	224         71         114         80         139           43         00         97         83         73         1           128         71         66         127         166	663         E22         L23         0         0         0         0           001         206         113         0         0         0         0         0           82         No         320         0         0         0         0         0					
IB Senden in MB/S 442	IB Senden in GB/s	del         1711         del           364         109         157         364           127         123         106         207           196         131         144         184	2168 44.5 0 109 177 345 109 198 228 10 0 0 0 10 0 10 0 0 10 0 10 10 0 10 0 1	647 9 6411 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	dx3         0         dx7           0         0         0         0         0           0         0         0         0         0         0           0         0         0         0         0         0         0	0         0	-90%
SATA MDS File Open in OPS/s	SATA MDS File Open in OPS/s	ds.4         1566         ds.4           266         120         010         137	2153 665 1777 353 6 0 0 0 0	6.8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	ds.4         0         0           0         0         0         0		
File Stat in OPS/s	File Stat in OPS/s	10         10 <sup>2</sup> 10 <sup>4</sup> 10 <sup>4</sup> 14         105         101         10           ds5         1754         455		60000000000000000000000000000000000000	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0			-80%
mkdir() in OPS/s	mkdir() in OPS/s	205         156         172         369           254         150         126         230           343         132         175         171	CAD         D02         27           BIT         BIC         IA           COD         S24					70%
unlink in OPS/s	Unlink in OPS/s	InfiniBand in MB/s Phase 1 (OSS für SATA)	Phase 1 Lustre (HDD) taurusoss1	Phase 2 Lustre (HDD), Gruppe 1 taurusoss5	Phase 2 Lustre (HDD), Gruppe 2 taurusoss9	Phase 2 Lustre (HDD), Gruppe 3 taurusoss13	Phase 2 Lustre (SDD)	-70%
Handan SATA		Tawar         Sin         Sin         Sin         Sint         S	COT14+1de3/111         Cot12+1de3/111         Cot12+1de3/111           COT24/1de3/111         771         771           COT24/1de3/101         11+43         11+43	rand vrite CETS4 Ide4/41 CETC0 Ide4/31 CETC0 Ide4/31 CETC0 Ide3/31	C011423 (dec5) 61 C01125 (dec5) 61 C01125 (dec5) 61 C01145 (dec5) 61 C01145 (dec5) 61 C01145 (dec5) 61	rand vrite 00742 (de12/10) 00772 (de12/2) 00772 (de12/2)	Control 1 (dulary 3) Control 1 (dulary 3) Control 1 (dulary 3) Control 1 (dulary 3)	-60%
bandvidth in MB/s	File Open in OPS/s	Phase 2 (OSS für SATA) Seiver 349 201 Sout 201	0073a (da5)61 720 00713 (da5)61 7 85 00713 (da5)61 7 85 00712 (da5)61 7 85 00723 (da6)61 7 85 00723 (da6)61 7 81	001776 (da5)41 001764 (da5)21 001764 (da5)10( 001764 (da5)10(	05167 (de.5)161 05167 (de.5)161 05167 (de.5)21 05127 (de.5)21	0776 (ds12/10) 0777 (ds12/0) 0776 (ds12/0) 07760 (ds12/0)	tavisions:22	
OPS/s in OPS/s 7961	File Stat in OPS/s	0006 	07722 (dek) 41 200 203 07764 (dek) 41 200 204 07764 (dek) 41 200 204 07764 (dek) 41 205 204 07764 (dek) 41 205 204	007145 (de 4/0) 00735 (de 2/0) 05700 (de 2/0) 00732 (de 2/10)	0 51 57 1 de 30 4 1 0 51 57 1 de 30 4 0 51 15 1 de 31 4 1 0 51 15 1 de 31 4 1 0 51 16 1 de 41 4 1	07762 (ds12/8) 07754 (ds12/4) 07714 (ds16/8) 07712 (ds16/4)	0 000 ( data) ( data) ( d) 000 ( data) ( d) 000	-50%
SATA OSTs Lesen in GB/s	unlink in OPS/s	00010 00012 00012 00013 00014	taurusoss2	taurusosső corristialagyal corristialagyal	taurusoss10 rend vrite 00134 (desg 3)	taurusoss14 orres veite orres (de12/2) orres (de12/2)	TAURIDOSLIS String Index par String Index par String Index par	
Schreiben in OPS/S 39	Weitere in OPS/s	00016 00016 00017 00018 00013	007129 (de/y31 20 023 007129 (de/y71 144 031 00728 (de/y71 144 031 00728 (de/y31 146 031 00728 (de/y31 146 031	007780 (dan) 11 00700 (dan) 71 00728 (dan) 20 00728 (dan) 20 00728 (dan) 20	00744 (de.9 7) 00757 (de.9 3) 00704 (de.9 3) 00704 (de.9 7)	00713 (4612/11) 00714 (4612/1) 00727 (4612/1) 0075e (4612/2)		-40%
	NetApp SATA	Phase 2 (OSS für SSD)	007166 (ds.2)364 01 233 00712 (ds.4)41 145 043 007167 (ds.2)71 115 764 007167 (ds.2)71 115 764	01749 (dag)71 01754 (dag)51 01754 (dag)11 01759 (dag)11	05758 (det)13) 05732 (det)1 05766 (det)9) 05766 (det)2)	05767 (ds1)/7 0374 (ds1)/7 05752 (ds10/5 03756 (ds10/1)		
	Dendwidth in NB/s	000 20 000 20 000 20	taurusoss3	taurusoss7	taurusoss11	taurus oss 15		-30%
	SATA OST		00726 (da37 2) 11 034 0076 (da37 2) 11 13 00716 (da37 2) 11 13 00712 (da7 4) 11 13 007120 (da47 4) 11 14 007120 (da4 4) 11	0071a (daty 4) 0071a) (daty 4) 007100 (daty 6) 00710a (daty 6)	com tro i da qu'al 05134 i da 714 i 05154 i da 719 i 05158 i da qu'al	00779 (ds10/10) 00708 (ds12/10) 00709 (ds12/10) 00714 (ds12/0)		20%
	Lesen in GB/s		00703 (akay 2) (44 200 00703 (akay 3) 111 276 00709 (akay 3) 116 204 00737 (akay 3) 116 204 00737 (akay 3) 116 204	00730 (36.5/3) 00759 (36.4/2) 00779 (36.9/10) 00770 (36.9/10) 00770 (36.9/2)	007100 (ab.0) 101 007140 (ab.0) 41 007140 (ab.0) 81 0070+0 (ab.0) 81 0070+0 (ab.0) 21	07750 (4612/0) 07757 (4612/4) 07753 (4612/0) 07754 (4612/0)		-20%
	Schreiben in OPS/s		donto rakayat taunusossa	orrad I da y el torrad I da y el ta unus oss8	omte Heleg 24 omte Heleg 261 taurus oss 12	omer (-610 s) omrs (-610 s) taurus oss 16		-10%
	NetApp SSD bandwidth in NB/s		rend vrite 0071a (daty3) 070 0071a (daty3) 030 00713 (daty3) 031 0710 (daty3) 031 0710 (daty3) 031	read vrite 007746 ide (7.7) 007750 ide (7.1) 00750 ide (7.1)	Contract (dasty 2) Octract (dasty 2) Octract (dasty 2) Octract (dasty 2)	rend vrite 007000 (de12)51 00700 (de12)11 00700 (de12)11 00700 (de12)11		
	OPS/s in OPS/s		00100 (dkg/3) 0 733 00100 (dkg/3) 0 1957 00100 (dkg/3) 0 001 00100 (dkg/3) 0 001	00734 (dag)1 00734 (dag)1 00734 (dag)1 00756 (dag)1 00756 (dag)1	00103 (des) 31 00143 (des) 31 00141 (des) 31 00141 (des) 31	0077+ (ds11/3) 07764 (ds13/1) 07750 (ds10/1) 07750 (ds10/1)		-0%
	SSD OST:		007126 (ds.4/2) 202 202 007126 (ds.7/4) 202 202 007126 (ds.7/4) 202 203	01749 (da 2/3) 01739 (da 2/3) 01746 (da 2/1)	001100 (de 7) 2 ( 001100 (de 7) 8) 001702 (de 7) 9)	00738 (de12/3) 00780 (de12/3) 0075c (de12/3)		





Michael Kluge, ZIH

Performance Analysis for I/O Activities (2)

- Amount of collected data:
  - -240 OSTs, 20 OSS servers, ...
  - Looking at stats and brw\_stats (how to store a histogram in database?)
  - $\rightarrow$  ~75.000 tables
  - about 1 GB of data per hour
- Analysis is supposed to cover 6 month
  - -4 TB data
  - No way to analyze this with any serial approach





#### **Analysis Process**

map operations to a set of tables

- modify all tables (align data, scale, aggregate over time)
- create a new table from an input set of tables (aggregation)
- -plot (time series, histograms)
- currently: single thread Haskell program
- Swift/T work in progress
  - 4 TB means 50 seconds on an 80 GB/s file system for the initial read
  - Swift/T has nice ways to describe data dependencies and to parallelize this kind of workflows (open for suggestions!)





- analysis of the raw performance data
  - take original data and create derived metrics
  - look at plots and histograms

- correlate metrics with user jobs
  - take job log and split jobs into classes
  - check for correlations between job classes and performance metrics





#### Metric Example: NetApp vs. OST Bandwidth (20s)

osts netpapps ECHNISCHE DRESDEN Center for Information Services & Michael Kluge, ZIH High Performance Computing

NetApp Bandwidth vs. OST bandwidth

## Metric Example: NetApp vs. OST Bandwidth (60s)

osts netpapps ECHNISCHE DRESDEN Center for Information Services & Michael Kluge, ZIH High Performance Computing

NetApp Bandwidth vs. OST bandwidth

#### Metric Example: NetApp vs. OST Bandwidth (10 minutes)





#### Metric Example: 3 Month bandwidth vs. IOPS



#### I/O and Job Correlation: Starting Point (Bandwidth)



1 week: allocated cores vs Lustre bandwidth

Michael Kluge, ZIH

High Performance Computing

#### **Building Job Classes**

- a class is defined by:
  - -user id, core count
  - -similar runtime
  - similar start time (time isolation)

**Require:** 
$$objs$$
 // set of objects  
1:  $num = 0$   
2: sort  $objs$  ordered by  $linearorder$   
3: insert  $objs[0]$  into  $clusters[num]$   
4: **for**  $i = 1; i < |objs|; i + + do$   
5: **if**  $d(objs[i-1], objs[i]) > d_{max}$  **then**  
6:  $num + +$   
7: **end if**  
8: insert  $objs[i]$  into  $clusters[num]$   
9: **end for**

- sort all jobs by runtime, split list at gaps
- split these groups again if they are not overlapping (long gap between start)
- challenges: runtime fluctuation, large run time variations





#### I/O and Job Correlation: Preliminary Result (1)



#### I/O and Job Correlation: Preliminary Result (1)



1 week: 77 jobs vs Lustre bandwidth

Michael Kluge, ZIH

20

High Performance Computing

#### I/O and Job Correlation: Correlation Factors

TECHNISCHE UNIVERSITÄT DRESDEN





Michael Kluge, ZIH



## **Open Topics**

- How to store a histogram in database?
- Other approaches than Swift/T
- Deal with the zeros
- Cache intermediate results





- it is possible to map anonymous performance data back to user jobs
- advance towards storage controller analysis (caches, backend bandwidth, ...)







