Storage-Monitoring und -Optimierung (nicht nur) für DL-Anwendungen

Christian Boehme Azat Khuziyakhmetov Sebastian Krey Hendrik Nolte

1. Oktober 2020

hpc@gwdg.de

Planung HPC Storage

Planung HPC Storage

- Speicherbedarf (aktuell und Zuwächse über Systemlaufzeit)
- Verteilung der Dateigrößen
- Burstbuffer und/oder hybride Storagepools
- Data on Metadata
- R/W Verteilung
- Verteilung sequentieller und zufälliger I/O
- Schreibzugriffe von einzelnen Knoten oder parallel von vielen Knoten

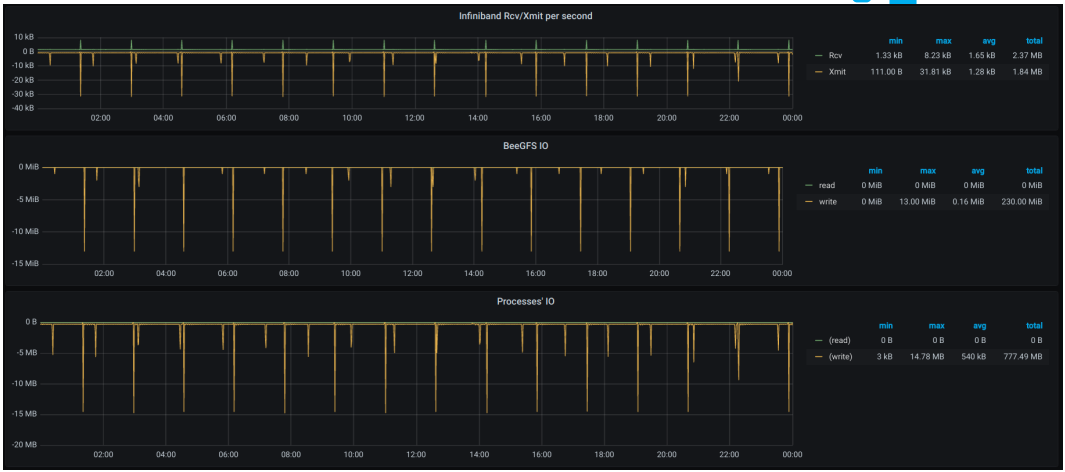
Beschaffung der Informationen

- Quota
- Policy Engines (Robinhood, Stratagem, etc.)
- I/O Statistiken von Storageservern
- I/O Statistiken von einzelnen Clients, Benutzern, Jobs
- Profiling von Jobs

Tools bei der GWDG

- Grafana und InfluxDB
- Telegraf
- BeeGFS Mon
- ProfitHPC (https://profit-hpc.de/)

ProfiT-HPC: Administrator view



ProfiT-HPC: Administrator view

In ProfiT-HPC werden Metriken von

- CPU
- Speicher
- GPUs
- Infiniband
- Dateisystem-IO (BeeGFS, lokale SSDs)
- Prozess-IO

gezeigt.

Es ist auch möglich, den Benutzern Empfehlungen in einem Bericht zu geben, wenn Werte außerhalb von definierten Schwellenwerten liegen

ProfiT-HPC: Beispiel eines Textberichts

JobID: 2837739
Used Nodes: 3
Requested Compute Units: 12

nodes	CPU usage cores	CPU usage total[%]	CPU usage hwm[%]	memory used mean[GiB]	memory used hwm[GiB]	memory alloc[GiB]
dge009	4	100	100	0.59	0.60	20.97

nodes	GPU ID [bus]	GPU usage mean[%]	GPU usage hwm[%]	GPU memory hwm[%]	CPU # proc total	CPU usage hwm[%]
dge009	02	1	4	4	2	50
	03	1	4	4	2	50

Per node IO and network. Sums and rates

nodes	Process IO rd(sum)	Process IO wr(sum)	Scratch FS rd(sum)	Scratch FS wr(sum)	Infiniband rcv[/s]	Infiniband snd[/s]
dge009	23.88 MiB	0 B	0 B	0 B	17.37 MiB	17.28 MiB

- All GPUs have low usage, which might be caused by misconfiguration of the application/job
- Some GPUs have more than 1 process using them at the same time. Check the balance between CPU processes and amount of used GPUs

IO Veränderungen durch Wandel in HPC

Traditionelle HPC Applikationen

- Oft Simulation
- Aus kleiner Modellbeschreibung werden große Datensätze
- Mixed R/W
- paralleler Schreibzugriff von vielen Knoten auf gleiche Dateien → File Locking wichtig
- Große Dateien
- Oft sequentieller IO

Grafana BeeGFS

Grafana BeeGFS GPU Cluster

Data Science/Data Analytics

- Aus großen Datenmengen kleines Modell extrahieren → read intensive
- Oft wiederholtes Lesen gleicher Daten für iteratives Lernen des Modells → kann von Caching profitieren
- Datensätze werden nach der Aufbereitung oft nur noch gelesen
- Erstellte Modelle sind relativ kleine Dateien, jeder Prozess schreibt eigenes Modell
- Training oft embarassingly parallel, wenig MPI
- Aufbereitung der Daten oft zufälliger I/O
- Daten zu groß für RAM also häufiges Öffnen, erneutes Lesen und Schließen gleicher Dateien (oft unnötig oft)

Konsequenzen

- Random I/O auf HDD basierten Storage führt zu langsamen I/O für alle
- Viele Metadatenoperation erzeugen hohe MDS Last $\rightarrow$ Langsames Dateisystem für alle
- Erneutes Lesen von unveränderten Dateien, die aus den Page Cache verdrängt wurden, erzeugt unnötigen Bandbreitenbedarf

Verbesserungansätze

Mögliche Ansätze zur Verbesserung

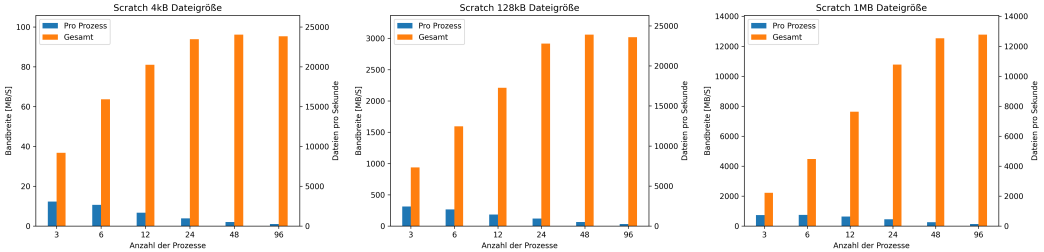
- Neben HPC Dateisystemen wie Lustre und BeeGFS auch wieder NFS mit fs-cache auf schnellen SSDs (siehe z.B. Nvidia DGX Referenzarchitektur)
- Lustre Persistent Client Cache
- Aufgabe der POSIX Kompatibilität und Nutzung von Objektstores über Schnittstellen wie S3, Intel DAOS, etc.
- Automatisches Tiering zwischen HDDs und SSDs
- Ad-hoc Dateisysteme (BeeGFS/Lustre On Demand, GekkoFS, etc.)
- NVMe over Fabrics (als Burstbuffer oder rohes Blockdevice, z.B. SPDK, Linux Kerneltreiber, diverse kommerzielle Lösungen)

Bei GWDG in Evaluation

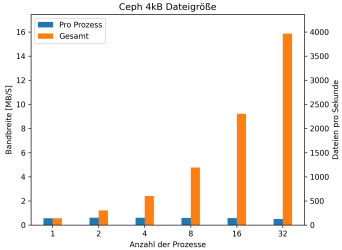
- CEPH S3 storage
- Re-Export von BeeGFS via NFS (read only) mit fs-cache
- Hybride Storagepools aus HDDs und SSDs mit ZFS und BeeGFS
- BeeGFS On Demand
- NVMe over Fabrics (SPDK)

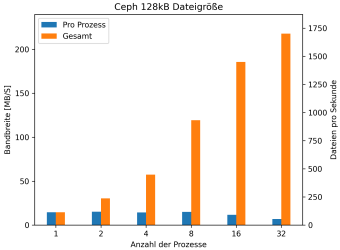
Benchmarks von BeeGFS und Ceph S3

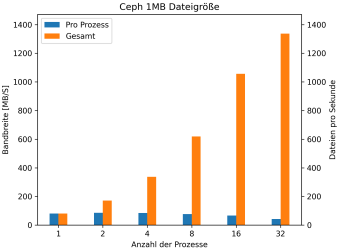
Benchmarks BeeGFS



Benchmarks Ceph







Fazit

Lessons learned

- S3 Leistung extrem vom Client abhängig.
- S3 pro Prozess langsamer als BeeGFS
- S3 Performance skaliert linear in der Anzahl der Prozesse bis zur Sättigung der Verbindung → Hohe Leistung möglich
- Gelegentliche hohe Antwortzeiten sorgen für gefühlt schlechte Leistung
- Tuning von Ceph S3 komplexer als BeeGFS
- Einfach zu benutzende Tools reduzieren die Einstiegsschwelle in das Job-Profiling und geben interessante Einblicke in das I/O Verhalten von Prozessen

Ausblick

- S3 Storage näher an HPC anbinden (konsistentere Latenzen, mehr Bandbreite)
- Vergleich verschiedener Objectstores (Ceph, MinIO, OpenIO)
- Ausbau des ProfitHPC Empfehlungssystems mit mehr Details zum Thema Storage
- Benutzerschulungen für AI/DL auf HPC Systemen
  - Minimierung von Metadatenoperationen
  - effizienter Small File I/O
  - Trennung Datenvorverarbeitung (I/O intensive) von Modelltraining/-auswertung (compute intensive)