Sizing Burst-Buffers Efficiently

Guillaume Aupy, Olivier Beaumont, Lionel Eyraud-Dubois

Inria & University of Bordeaux, France

Scheduling Workshop, Berkeley, June 2018

Motivation

IO congestion in HPC systems:

- HPC applications are generating lots of data for PFS.
- Idea is to use a buffer when the I/O bandwidth is fully occupied
- The buffer can be emptied at a later time.

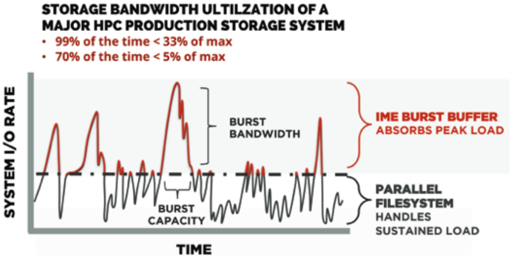


Figure: Burst-buffers to absorb IO peaks

Source: DDN ad material.

Some use of Burst-Buffers

Historically, Burst-Buffers were attached to IONodes (ION), used as buffers when the I/O Bandwidth was not enough (Gordon@SDSC).

But many other possible uses:

- For temporary data that may eventually not be needed (e.g. fault-tolerance)
- For intermediate data (e.g. BigData on HPC machine, In-situ/In-transit)
- For other uses?

Some use of Burst-Buffers

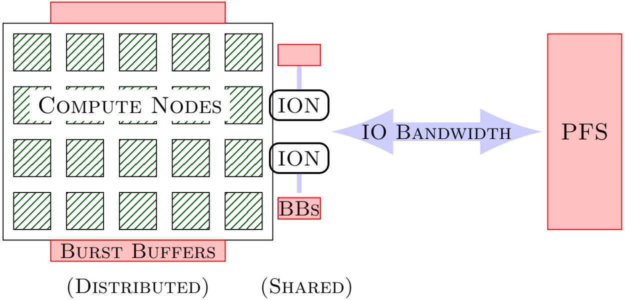
Historically, Burst-Buffers were attached to IONodes (ION), used as buffers when the I/O Bandwidth was not enough (Gordon@SDSC).

But many other possible uses:

- For temporary data that may eventually not be needed (e.g. fault-tolerance)
- For intermediate data (e.g. BigData on HPC machine, In-situ/In-transit)
- For other uses?

How do we choose the *right* amount of Burst-Buffers for each use?

A few questions to answer (I)



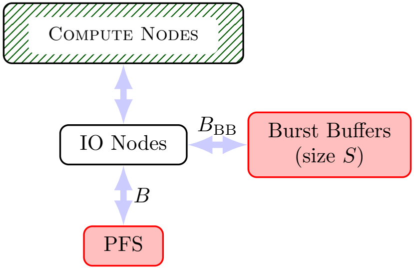
Application Models:

- Compute and I/O behaviors, buffer needs?
- Performance model of application?

Distributed Buffers:

- How to partition the buffers amongst applications?
- Location of data?

A few questions to answer (II)



Centralized buffers for I/O management:

- What bandwidth B_{BB}?
- What size S?
- What filling/emptying policy?

Filling/Emptying policy

Filling policies

- Use as a **Cache** to PFS.
  - Pro: More efficient, lower latency
  - Cons: If SSD-based, limited write-life
- Use as a **Buffer** when too many simultaneous I/O calls

Filling/Emptying policy

Filling policies

- Use as a **Cache** to PFS.
  - Pro: More efficient, lower latency
  - Cons: If SSD-based, limited write-life
- Use as a **Buffer** when too many simultaneous I/O calls

Emptying policies

- When some I/O bandwidth is available, empty as much as possible.
- When some I/O bandwidth is available, AND Burst-Buffers are at least $T\%$ full, empty as much as possible.

Application model

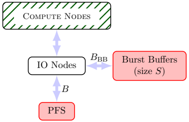
We consider a unit time characteristic of the system.

Applications: At any time unit, application $\mathcal{A}_i$ sends data:

- with probability p_i
- at bandwidth b_i.

Machine is characterized by:

- The Burst Buffer size S
- Its expected IO load: $\textsc{ExpectedLoad} = \sum_i p_i b_i$;
- Its bandwidth to PFS: B



What bandwidth B_{BB}?

X_i: random variable indicating whether $\mathcal{A}_i$ is sending I/Os.
$\rightarrow X_i = 1$ with proba p_i and 0 with $1 - p_i$.

$$\text{Instant bandwidth } X = \sum_i b_i X_i$$

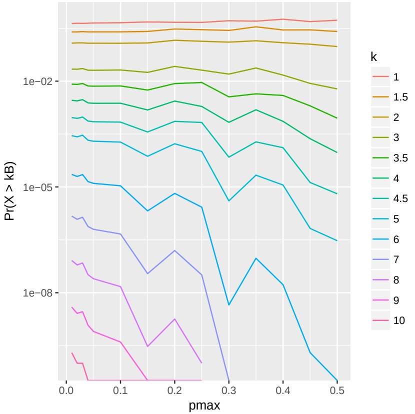
What bandwidth B_{BB}?

X_i: random variable indicating whether $\mathcal{A}_i$ is sending I/Os.
$\rightarrow X_i = 1$ with proba p_i and 0 with $1 - p_i$.

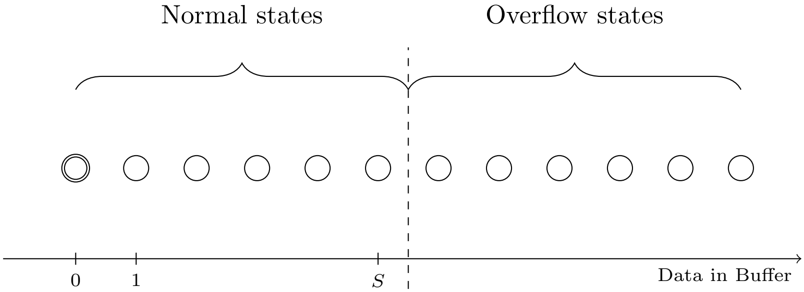
$$\text{Instant bandwidth } X = \sum_i b_i X_i$$

Simulation setup: we fix $p_{\max}$.

- While $\sum_i p_i b_i < B$:
- Create a new application with p_i (resp. b_i) chosen uniformly at random in $[0, p_{\max}]$ (resp. $[0, B]$).

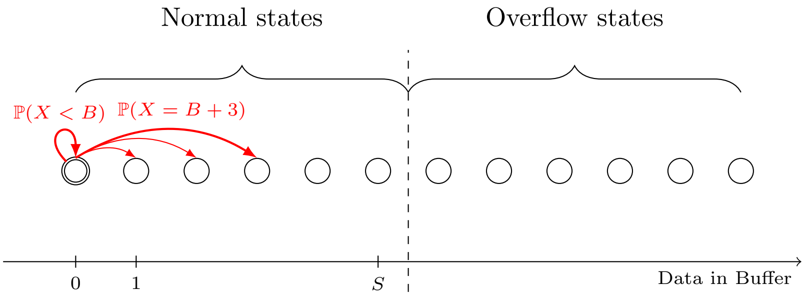


Platform model: when buffer full, stall all applications for one time unit



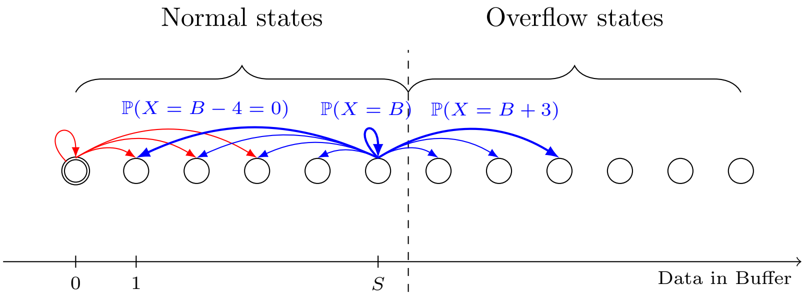
Modeling with Markov chains

Platform model: when buffer full, stall all applications for one time unit



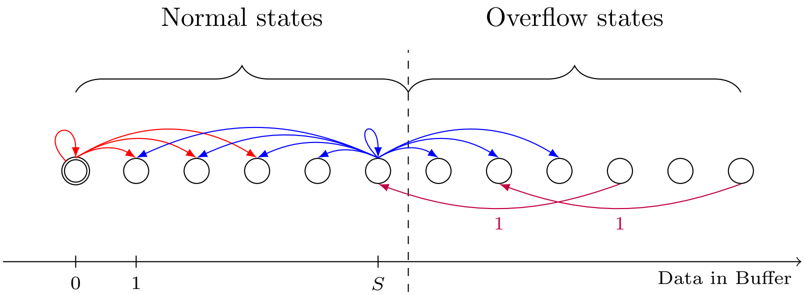
Modeling with Markov chains

Platform model: when buffer full, stall all applications for one time unit

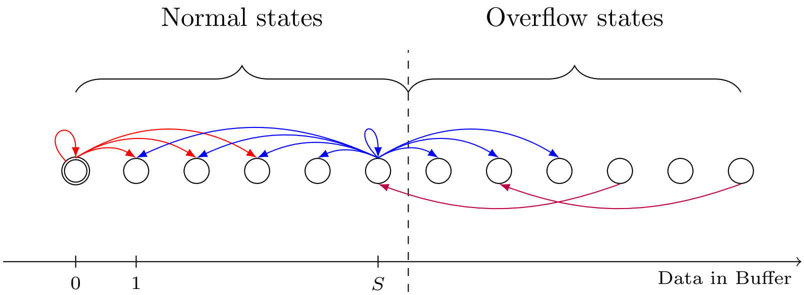


Modeling with Markov chains

Platform model: when buffer full, stall all applications for one time unit



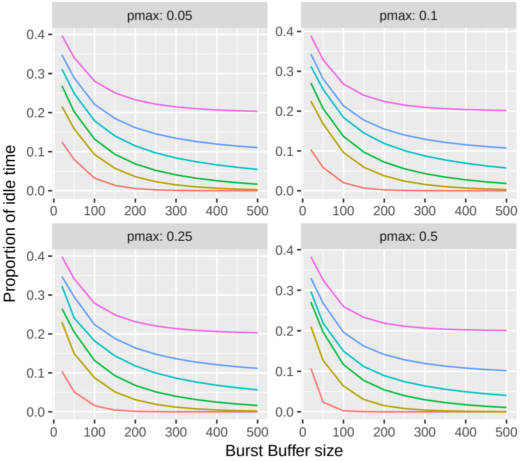
Platform model: when buffer full, stall all applications for one time unit



Results

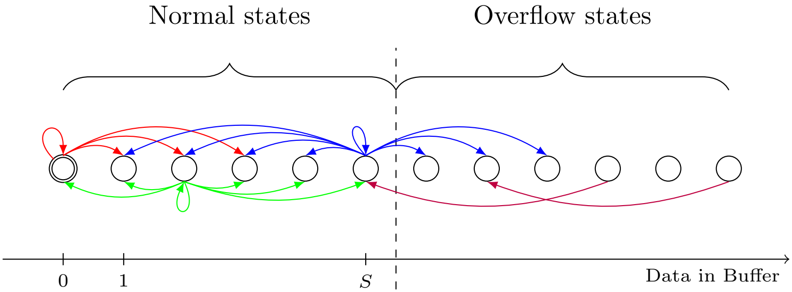
- Aperiodic and irreducible MC: unique stationary probability π
- Fraction of time spent idle is $\sum_{s \in \text{overflow states}} \pi_s$

Proportion of idle time as a function of buffer size, for different values of $p_{\max}$ and stress $\alpha = \frac{\text{EXPECTEDLOAD}}{B}$ $(B = 100)$

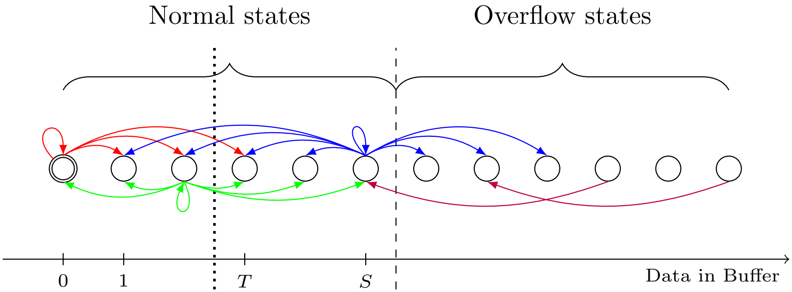


What policy?

Lazy Emptying [Cluster 2017]: Only empty the burst buffer when its load reaches a threshold T .

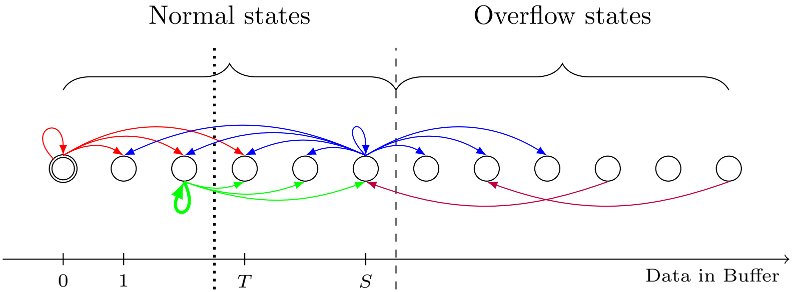


Lazy Emptying [Cluster 2017]: Only empty the burst buffer when its load reaches a threshold T .

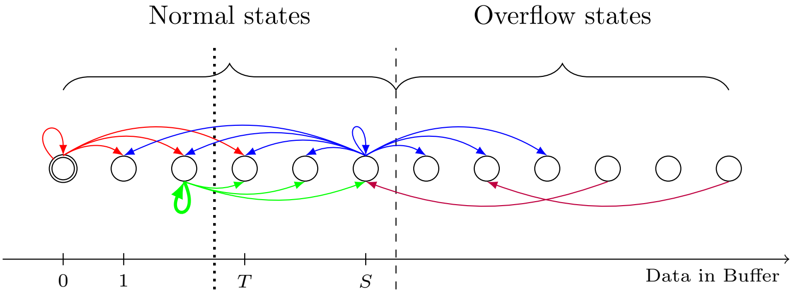


WHAT POLICY?

Lazy Emptying [Cluster 2017]: Only empty the burst buffer when its load reaches a threshold T .

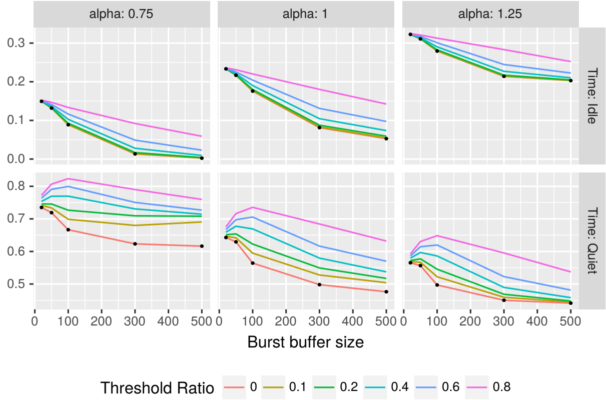


Lazy Emptying [Cluster 2017]: Only empty the burst buffer when its load reaches a threshold T .



- Still unique stationary distribution π^{Lazy}
- Quiet time (no data sent from buffer):

$$\sum_{s \in \text{normal state}} \pi_s^{\text{Lazy}} \cdot \sum_{t \geq s} P^{\text{Lazy}}(s,t)$$



Lazy Emptying: Only empty the burst buffer when its load reaches a threshold T (black dots are for $T = 0$, $p_{\max} = 0.1$).

Comparison with a different model

Pseudo-periodic model: application $\mathcal{A}_i$

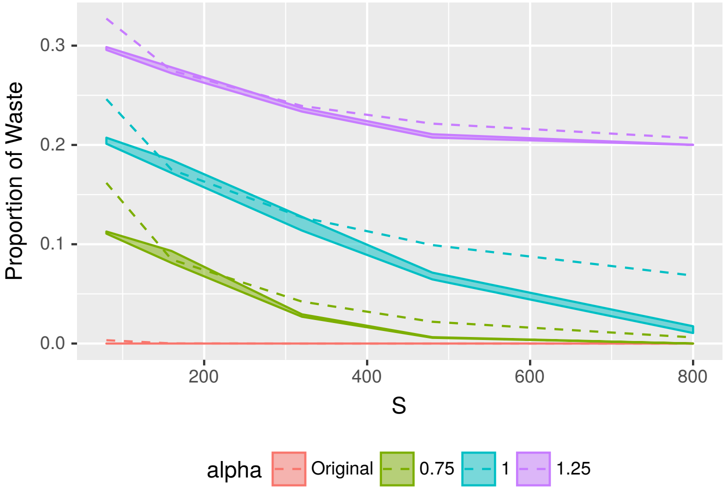
- has a **pseudo-period** d_i
- **sends data** at bandwidth b_i during $p_i d_i \pm 25\%$
- **computes** during $(1 - p_i)d_i \pm 25\%$

Evaluated on applications from APEX data set

[LANL Tech. Report, 2016]

Workflow	EAP	LAP	Silverton	VPIC
Number of Instances	13	4	2	1
b_i (GB/s)	160	80	160	160
d_i Period (s)	5671	12682	15005	4483
Checkpoint time (s)	20	25	280	23,4
$p_i(\times 10^{-3})$	3.51	1.97	18.7	5.11

Comparison with a different model



- Increase stress α by scaling up p_i values
- Close behavior despite very different model

Summary

- Tractable model for dimensioning Burst Buffers
  - Which size for a given stress?
  - Emptying threshold: 20-40% is a reasonable choice
  - Validated against a different model
- Further questions
  - Other application models (maybe not Markovian)
  - Characterization of I/O patterns
  - Improve platform model (congestion, distributed BB)
- Open for criticism, remarks, suggestions, and collaborations!