Sizing and Partitioning Strategies for Burst-Buffers to Reduce Contention

Guillaume Aupy, Olivier Beaumont, Lionel Eyraud-Dubois
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.

Burst-buffers to absorb IO peaks

Source: DDN ad material.
Application Context

Main transfer source in large HPC applications: checkpoints (⇒ predictable)

Possible usages for Burst-buffers:

► accelerate one application by caching writes
► hide contention coming from several applications writing at the same time

And with BigData applications coming:

► prefetch input data
► cache for temporary data
Applications run on Compute Nodes
- Placement already done

Two buffer management policies:
- **STATIC**: size $S_k$ allotted to application $A_k$ for its lifetime
- **DYNAMIC**: size devoted to $A_k$ can change over time
Application model

Set of applications \( \mathcal{A}_k \) running independently on the platform:

- with release date \( r_k \), read and write bandwidth \( b^r_k \) and \( b^w_k \)
- consisting of \( n_k \) phases (without overlap):
  - Read a volume of \( R^k_i \) input data starts at \( t^r_{i,k} \)
  - Compute for \( l^k_i \) amount of time starts at \( t^c_{i,k} \)
  - Write a volume of \( W^k_i \) output data starts at \( t^w_{i,k} \)

- No overlap: data available from the start, but two phases do not fit in memory

Earliest possible completion time:

\[
C^\text{min}_k = r_k + \sum_{i=1}^{n_k} \left( \frac{R^k_i}{b^r_k} + \frac{W^k_i}{b^w_k} \right)
\]

Stretch of \( \mathcal{A}_k \):

\[
s(\mathcal{A}_k) = \frac{C_k}{C^\text{min}_k}
\]
Execution model

\[ \mathcal{A}_k \rightarrow f_k^r + i_k^r \]

Completely determined by the amount of data transferred at each event.

Dominant schedules: all transfer rates constant between time events.

Rate: \( b_k^r \)

Time: \( t \)

- \( t_{k,1}^r \): read
- \( t_{k,1}^c \): compute
- \( t_{k,1}^w \): write
- \( t_{k,2}^r \): read
- \( t_{k,2}^c \): compute
Execution model

Dominant schedules: all transfer rates constant between time events

Completely determined by the amount of data transferred at each event
Execution model

Dominant schedules: all transfer rates constant between time events

Completely determined by the amount of data transferred at each event
Execution model

\[ \mathcal{A}_k \xrightarrow{f_k^r + i_k^r} f_k \]

Dominant schedules: all transfer rates constant between time events.

Completely determined by the amount of data transferred at each event.
Dominant schedules: all transfer rates constant between time events
Execution model

Dominant schedules: all transfer rates constant between time events
Execution model

Dominant schedules: all transfer rates constant between time events
Execution model

Dominant schedules: all transfer rates constant between time events
Completely determined by the amount of data transferred at each event
Problem formulations – both Static and Dynamic

**Static-Buffer-Size(\(\rho\)) and Dynamic-Buffer-Size(\(\rho\))**

Given \(n\) applications \((A_k)\), and a stretch limit \(\rho\), minimize the total buffer size \(S\) necessary to achieve stretch \(\rho\).

**Static-Stretch(\(S\)) and Dynamic-Stretch(\(S\))**

Given \(n\) applications \((A_k)\), and a buffer size \(S\), minimize the maximum stretch over all applications.

**Results**

- **X-Stretch(0)** is NP-complete
- **X-Buffer-Size(\(\rho\))** is NP-complete for \(1 < \rho \leq 2\)
- **Static-Stretch(\(S\))** is NP-complete for all \(S\)
Problem formulations – both Static and Dynamic

Static-Buffer-Size($\rho$) and Dynamic-Buffer-Size($\rho$)
Given $n$ applications ($A_k$), and a stretch limit $\rho$, minimize the total buffer size $S$ necessary to achieve stretch $\rho$.

Static-Stretch($S$) and Dynamic-Stretch($S$)
Given $n$ applications ($A_k$), and a buffer size $S$, minimize the maximum stretch over all applications

Results

- $X$-Stretch(0) is NP-complete
- $X$-Buffer-Size($\rho$) is NP-complete for $1 < \rho \leq 2$
- Static-Stretch($S$) is NP-complete for all $S$
- $X$-Buffer-Size(1) can be solved in polynomial time
Scheduling a single application

Aiming for stretch 1 ($C_k = C_k^{\text{min}}$) fixes the values of $t_{i,k}^*$

Size of data

$W_k^\infty(t)$, written to PFS if $B_w = \infty$
Scheduling a single application

Aiming for stretch 1 \((C_k = C_k^{\min})\) fixes the values of \(t_{i,k}^*\)

\[
\begin{align*}
W_k^\infty(t), & \text{ written to PFS if } B_w = \infty \\
-w_k(t), & \text{ written to PFS if } B_w < b_k^w
\end{align*}
\]
Scheduling a single application

Aiming for stretch 1 ($C_k = C_k^{\text{min}}$) fixes the values of $t_{i,k}^*$.
Scheduling a single application

Aiming for stretch 1 ($C_k = C_k^{\text{min}}$) fixes the values of $t_{i,k}^*$.
Linear Programming Formulation

Consider all events \( \{e_l\} = \{t_{k,i}^*\} \): variables \( w_k^l \) and \( r_k^l \), variable \( S \) for buffer size

Minimize \( S \) subject to:

\[
\forall l, \quad \left( W_k^\infty(e_l) - w_k^l + r_k^l - R_k^\infty(e_l) \right) \leq S
\]

Data stored in buffer

- Output data
- Input data

\[
\forall l, \quad w_k^l \leq w_k^{l+1}
\]

Amount of data is non-decreasing

\[
\forall l, \quad w_k^l \leq W_k^\infty(e_l)
\]

Can not write more than app. sends

\[
\forall l, \quad w_k^{l+1} - w_k^l \leq B_w(e_{l+1} - e_l)
\]

Can not write faster than PFS accepts

\[
\forall l, \quad r_k^l \leq r_k^{l+1}
\]

Amount of data is non-decreasing

\[
\forall l, \quad r_k^l \geq R_k^\infty(e_l)
\]

Must read at least what app. needs

\[
\forall l, \quad r_k^{l+1} - r_k^l \leq B_r(e_{l+1} - e_l)
\]

Can not read faster than PFS provides

\[
\forall l, \quad w_k^{l+1} + r_k^{l+1} \leq B(e_{l+1} - e_l)
\]

Total bandwidth not exceeded
Linear Programming Formulation

Consider all events \( \{e_l\} = \{t_{k,i}^*\} \): variables \( w_k^l \) and \( r_k^l \), variable \( S \) for buffer size

Minimize \( S \) subject to:

\[
\forall l, \quad W_k^\infty(e_l) - w_k^l + r_k^l - R_k^\infty(e_l) \leq S
\]

Data stored in buffer

\[
\forall l, \quad w_k^l \leq w_k^{l+1}
\]

Amount of data is non-decreasing

\[
\forall l, \quad w_k^l \leq W_k^\infty(e_l)
\]

Can not write more than app. sends

\[
\forall l, \quad w_k^{l+1} - w_k^l \leq B_w(e_{l+1} - e_l)
\]

Can not write faster than PFS accepts

\[
\forall l, \quad r_k^l \leq r_k^{l+1}
\]

Amount of data is non-decreasing

\[
\forall l, \quad r_k^l \geq R_k^\infty(e_l)
\]

Must read at least what app. needs

\[
\forall l, \quad r_k^{l+1} - r_k^l \leq B_r(e_{l+1} - e_l)
\]

Can not read faster than PFS provides

\[
\forall l, \quad w_k^{l+1} - w_k^l + r_k^{l+1} - r_k^l \leq B(e_{l+1} - e_k)
\]

Total bandwidth not exceeded
For several applications

Compute all events \( \{e_l\} = \{t_{k,i}^*\} \) for all applications \( A_k \), in increasing order

Variables: \( w_k^l \) and \( r_k^l \), \( S_k^l \) (buffer size of \( A_k \) at event \( e_l \))

Minimize \( S \) subject to:

\[
\forall l, \quad \sum_k S_k^l \leq S \quad \text{Total buffer size}
\]

\[
\forall l, k \quad W_k^\infty(e_l) - w_k^l + r_k^l - R_k^\infty(e_l) \leq S_k^l \quad \text{Data stored in buffer}
\]

\[
\forall k, l \in \mathcal{I}_k \quad S_k^l = S_k \quad \text{Static constraint}
\]

\[
\forall l, k \quad w_k^l \leq w_k^{l+1} \quad \text{Amount of data is non-decreasing}
\]

\[
\forall l, k \quad w_k^l \leq W_k^\infty(e_l) \quad \text{Can not write more than app. sends}
\]

\[
\forall l, \quad \sum_k w_k^{l+1} - w_k^l \leq B_w(e_{l+1} - e_l) \quad \text{Can not write faster than PFS accepts}
\]

\[
\cdots \cdots \quad \text{[Read constraints]}
\]
Settings: applications from LANL Computing Center

<table>
<thead>
<tr>
<th>Workflow</th>
<th>EAP</th>
<th>LAP</th>
<th>Silverton</th>
<th>VPIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>65</td>
<td>21</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td># cores (1000)</td>
<td>16</td>
<td>4</td>
<td>32</td>
<td>30</td>
</tr>
<tr>
<td>Ckpt size (GB)</td>
<td>3,200</td>
<td>2,000</td>
<td>44,800</td>
<td>3,750</td>
</tr>
<tr>
<td>Walltime (hours)</td>
<td>16</td>
<td>4</td>
<td>32</td>
<td>30</td>
</tr>
</tbody>
</table>

Platform characteristics:
- 96,000 cores
- \( B = 160\)GB/s
- \( b = 20\)MB/s per core
- Period \( P = \sqrt{2C \frac{\mu}{\#\text{nodes}}} \)
- 5 years \( \leq \mu \leq 50 \) years

Generating instances

- Fix load \( \rho \) to 20, 50 or 80%
- Pick 30 applications according to Frequency
- Scheduling them FIFO yields release times
- Compute maximum average bandwidth requirement
- Scale checkpoint size to achieve max. load \( \rho \)
Settings

Greedy strategy

- $A_k$ is active if it is in a write-phase or has data in its buffer
- Bandwidth to the PFS is equally shared between all active applications
- $A_k$ sends data at maximum rate $b^w_k$ if its buffer has available space
- Otherwise, $A_k$ is limited to the available bandwidth to PFS

Methodology

- For each instance, compute $S^{OPT}$ and $S_k^{OPT}$ with $\text{STATIC-BUFFER-SIZE}(1)$
- $S$ varies between 0 and $3S^{OPT}$, scale $S_k$ accordingly
- Apply Greedy strategy, compute maximum stretch
Results

MTBF: 5
MTBF: 10
MTBF: 25
MTBF: 50

Buffer size
Maximum stretch
IO load
0.2
0.5
0.8
Overhead of **Static** vs **Dynamic**

Compare the optimal solutions with and without the **Static** constraint

<table>
<thead>
<tr>
<th>Load \ MTBF</th>
<th>5 y</th>
<th>10 y</th>
<th>25 y</th>
<th>50 y</th>
</tr>
</thead>
<tbody>
<tr>
<td>20%</td>
<td>1.32</td>
<td>1.31</td>
<td>1.42</td>
<td>1.67</td>
</tr>
<tr>
<td>50%</td>
<td>1.33</td>
<td>1.28</td>
<td>1.26</td>
<td>1.47</td>
</tr>
<tr>
<td>80%</td>
<td>1.23</td>
<td>1.26</td>
<td>1.25</td>
<td>1.35</td>
</tr>
</tbody>
</table>

Static constraint yields roughly 30% overhead
Conclusions

Remarks about model

- Previous paper [IPDPS'18]: random application behavior, Markov Chain modeling
- Here: all phases of applications are known, Linear Programming optimal solution

Perspectives

- Within this model: efficient strategies for small buffer size
- Extend the model: data reuse, temporary checkpoint data