Algorithmes d'ordonnancement et schémas de résilience pour les pannes et les erreurs silencieuses

Aurélien Cavelan  <aurelien.cavelan@unibas.ch>
ENS de Lyon et Inria, France

January 31, 2017
## Top500 List

<table>
<thead>
<tr>
<th>Rank</th>
<th>System</th>
<th>Cores</th>
<th>Rmax (TFlop/s)</th>
<th>Rpeak (TFlop/s)</th>
<th>Power (kW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sunway TaihuLight - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, NRPC, National Supercomputing Center in Wuxi, China</td>
<td>10,649,600</td>
<td>93,014.6</td>
<td>125,435.9</td>
<td>15,371</td>
</tr>
<tr>
<td>2</td>
<td>Tianhe-2 (MilkyWay-2) - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P, NUDT, National Super Computer Center in Guangzhou, China</td>
<td>3,120,000</td>
<td>33,862.7</td>
<td>54,902.4</td>
<td>17,808</td>
</tr>
<tr>
<td>3</td>
<td>Piz Daint - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect, NVIDIA Tesla P100, Cray Inc, Swiss National Supercomputing Centre (CSCS), Switzerland</td>
<td>361,760</td>
<td>19,590.0</td>
<td>25,326.3</td>
<td>2,272</td>
</tr>
<tr>
<td>4</td>
<td>Gyoukou - ZettaScaler-2.2 HPC system, Xeon D-1571 16C 1.3GHz, Infiniband EDR, PEZy-SC2 700Mhz, ExaScaler, Japan Agency for Marine-Earth Science and Technology, Japan</td>
<td>19,860,000</td>
<td>19,135.8</td>
<td>26,192.0</td>
<td>1,350</td>
</tr>
<tr>
<td>5</td>
<td>Titan - Cray XK7, Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x, Cray Inc, DOE/SC/Oak Ridge National Laboratory, United States</td>
<td>560,640</td>
<td>17,590.0</td>
<td>27,112.5</td>
<td>8,209</td>
</tr>
<tr>
<td>6</td>
<td>Sequoia - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom, IBM DOE/NNSA/LLNL, United States</td>
<td>1,572,864</td>
<td>17,173.2</td>
<td>20,132.7</td>
<td>7,890</td>
</tr>
<tr>
<td>7</td>
<td>Trinity - Cray XC40, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect,</td>
<td>979,968</td>
<td>14,137.3</td>
<td>43,902.6</td>
<td>3,844</td>
</tr>
</tbody>
</table>
Outline

I. Failures

II. Silent Errors
I. Failures
Mean Time Between Failures (MTBF)

Consider one processor (e.g. in your laptop):

- **MTBF = 100 years**
- (Almost) no failures in practice ...

**Theorem.**

MTBF decreases linearly with the number of processors.

- 36500 processors
  - **MTBF = 1 day**
  - A failure every day on average!

Large simulations can execute for weeks at a time.
A petascale computer

- $400m^2$
- 17.59 PetaFlops
- 693.6 TiB of RAM

**Titan has 37376 processors and GPUs and $\approx 1$ day MTBF.**
Fail-Stop Errors

Failures proportional to number of processors

- 2013: **Preproduction** Blue Waters requires repairs $\approx 4$ hours
- 2014: Titan ($37,376$ processors) loses a node every $\approx 1.5$ days
- 2015: Blue Waters ($26,868$ processors) loses $\approx 2$ nodes per day

Characteristics

- Component failure (node, network, power, ...)
- Application fails and data is lost
An Inconvenient Truth

Top ranked supercomputers in the US (June 2017)

<table>
<thead>
<tr>
<th>Rank</th>
<th>Name</th>
<th>Laboratory</th>
<th>Technology</th>
<th>Processors</th>
<th>PFlops/s</th>
<th>MTBF</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Titan</td>
<td>ORNL</td>
<td>Cray XK7</td>
<td>37,376</td>
<td>17.59</td>
<td>≈ 1 day</td>
</tr>
<tr>
<td>6</td>
<td>Sequoia</td>
<td>LLNL</td>
<td>BG/Q</td>
<td>98,304</td>
<td>17.17</td>
<td>≈ 1 day</td>
</tr>
<tr>
<td>8</td>
<td>Cori</td>
<td>LBNL</td>
<td>Cray XC40</td>
<td>11,308</td>
<td>14.01</td>
<td>≈ 1 day</td>
</tr>
<tr>
<td>11</td>
<td>Mira</td>
<td>ANL</td>
<td>BG/Q</td>
<td>49,152</td>
<td>8.59</td>
<td>≈ 1 day</td>
</tr>
</tbody>
</table>

The first exascale computer ($10^{18}$ FLOPS) is expected by 2020:

- Larger processors count: millions of processors
- MTBF is expected to drop dramatically
- Down to the hour or even worse

Coping with failures:

- Make applications more fault tolerant!
- Design better resilience techniques!
An Inconvenient Truth

Top ranked supercomputers in the US (June 2017)

<table>
<thead>
<tr>
<th>Rank</th>
<th>Name</th>
<th>Laboratory</th>
<th>Technology</th>
<th>Processors</th>
<th>PFlops/s</th>
<th>MTBF</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Titan</td>
<td>ORNL</td>
<td>Cray XK7</td>
<td>37,376</td>
<td>17.59</td>
<td>≈ 1 day</td>
</tr>
<tr>
<td>6</td>
<td>Sequoia</td>
<td>LLNL</td>
<td>BG/Q</td>
<td>98,304</td>
<td>17.17</td>
<td>≈ 1 day</td>
</tr>
<tr>
<td>8</td>
<td>Cori</td>
<td>LBNL</td>
<td>Cray XC40</td>
<td>11,308</td>
<td>14.01</td>
<td>≈ 1 day</td>
</tr>
<tr>
<td>11</td>
<td>Mira</td>
<td>ANL</td>
<td>BG/Q</td>
<td>49,152</td>
<td>8.59</td>
<td>≈ 1 day</td>
</tr>
</tbody>
</table>

The first exascale computer \((10^{18} \text{ FLOPS})\) is expected by 2020:

- Larger processors count: millions of processors
- **MTBF is expected to drop dramatically**
- Down to the **hour** or even worse

Coping with failures:

- Make applications more fault tolerant!
- **Design better resilience techniques!**
Coping with Fail-Stop Errors

Periodic checkpoint, rollback, and recovery:

- **Coordinated checkpointing (the platform is a giant macro-processor)**
- Assume instantaneous interruption and detection.
- Rollback to last checkpoint and re-execute.
Coping with Fail-Stop Errors

Periodic checkpoint, rollback, and recovery:

- Coordinated checkpointing (the platform is a giant macro-processor)
- Assume instantaneous interruption and detection.
- Rollback to last checkpoint and re-execute.
Coping with Fail-Stop Errors

Periodic checkpoint, rollback, and recovery:

- Coordinated checkpointing (the platform is a giant macro-processor)
- Assume instantaneous interruption and detection.
- Rollback to last checkpoint and re-execute.
Optimal Checkpoint Interval
Minimize Expected Execution Time

- $T$: Pattern length
- $C$: Checkpoint time
- $R$: Recovery time

\[ E(T) = P_{no-error} \cdot (T + C) \]

(no error)
Minimize Expected Execution Time

- \( T \): Pattern length
- \( C \): Checkpoint time
- \( R \): Recovery time

For no error case:

\[ E(T) = P_{no-error} \cdot (T + C) \]

For fail-stop error case:

\[ E(T) = P_{error} \cdot \left( E_{\text{lost}} + R + E(T) \right) \]
Optimization

› Choose fault-model
› Minimize $\mathbb{E}(T)$ for $T$

... 

Theorem. [Young 1974, Daly 2006]

$$T^{opt} = \sqrt{2\mu C}$$

› $C$: checkpoint cost
› $\mu$ (MTBF): Mean Time Between Failures
Extension: Multiple Levels of Checkpoints

Now, suppose that multiple types of checkpoints are available, e.g.

- Parallel File System (PFS)
- Local memory/SSD
- Partner copy/XOR

We can use synchronized checkpointing:

\[ \begin{array}{c|c|c|c|c|c|c|c} 
C_1 & C_2 & C_3 & C_1 & C_2 & C_1 & C_1 & C_2 & C_3 \\
\end{array} \]

Synchronized checkpointing

- \( k \) levels of checkpoints
- An error at level \( i \) kills all checkpoints \( C_j < C_i \)
Extension: Multiple Levels of Checkpoints

Now, suppose that multiple types of checkpoints are available, e.g.
- Parallel File System (PFS)
- Local memory/SSD
- Partner copy/XOR

We can use **synchronized** checkpointing:

![Diagram of synchronized checkpointing]

Synchronized checkpointing

- $k$ levels of checkpoints
- An error at level $i$ kills all checkpoints $C_j < C_i$
Generalization to $k$ Levels

**Theorem (single-level) [Young, Daly]:**

Optimal pattern length: $T^{\text{opt}} = \sqrt{\frac{2C}{\lambda}}$

**Theorem (multi-level):**

Optimal pattern length: $T^{\text{opt}} = \sqrt{\frac{\sum_{\ell=1}^{k} N_{\ell}^{\text{opt}} C_{\ell}}{\frac{1}{2} \sum_{\ell=1}^{k} \frac{\lambda_{\ell}}{N_{\ell}^{\text{opt}}}}}$

Optimal #chkpts at level $\ell$: $N_{\ell}^{\text{opt}} = \sqrt{\frac{\lambda_{\ell}}{C_{\ell}} \cdot \frac{C_{k}}{\lambda_{k}}}, \ \forall \ell = 1, \ldots, k$
II. Silent Errors

a.k.a. Silent Data Corruptions
**Silent Data Corruptions**

<table>
<thead>
<tr>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit flip (Disk, RAM, Cache, Bus, ...)</td>
</tr>
<tr>
<td>Problem: detection latency, wrong results</td>
</tr>
</tbody>
</table>

**Number of errors proportional to area and circuit design**

- **2002:** **Unprotected address bus** ASCI Q at Los Alamos National Laboratory could not run more than one hour
- **2003:** **No ECC** Virginia Tech 1,100 Apple Power Mac G5 supercomputer could not boot
- **2010:** **ECC protected** Jaguar saw 350 bit-flips/min
- **2010:** **ECC protected** Jaguar saw 1 double-bit error/day
- **2014:** Titan: **reported** > 1 Double Bit Error per week

Silent errors can no longer be ignored!
Methods for Detecting Silent Errors

General-purpose approaches

- Replication [Fiala et al. 2012] or triple modular redundancy and voting [Lyons and Vanderkulk 1962]

Application-specific approaches

- Algorithm-based fault tolerance (ABFT): checksums in dense matrices Limited to one error detection and/or correction in practice [Huang and Abraham 1984]
- Partial differential equations (PDE): use lower-order scheme as verification mechanism [Benson, Schmit and Schreiber 2014]
- Preconditioned conjugate gradients (PCG): orthogonalization check every $k$ iterations, re-orthogonalization if problem detected [Sao and Vuduc 2013, Chen 2013]

Data-analytics approaches

- Dynamic monitoring of HPC datasets based on physical laws (e.g., temperature limit, speed limit) and space or temporal proximity [Bautista-Gomez and Cappello 2014]
- Time-series prediction, spatial multivariate interpolation [Di et al. 2014]
Coping with Fail-Stop and Silent Errors

What is the optimal checkpointing period?
Resilience Patterns

Starting with base pattern

Simple pattern (Young-Daly)

Adding verified memory checkpoints

Pattern with $n$ segments
Resilience Patterns

Adding intermediate verifications between memory checkpoints

Segment $w_i$ has $m_i$ chunks

Putting everything together

Full pattern
### Theorems

**Our contributions:**

<table>
<thead>
<tr>
<th>Pattern</th>
<th>(T^*)</th>
<th>(n^*)</th>
<th>(m^*)</th>
<th>(H^*(\text{Pattern}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P_D)</td>
<td>(\sqrt{\frac{V^*+C_M+C_D}{\lambda^s+\lambda^f}})</td>
<td>-</td>
<td>-</td>
<td>(2\sqrt{\left(\frac{\lambda^s+\lambda^f}{2}\right)(V^*+C_M+C_D)})</td>
</tr>
<tr>
<td>(P_{DV^*})</td>
<td>(\sqrt{\frac{\frac{m^<em>V^</em>+C_M+C_D}{\lambda^s+\lambda^f}}{\left(1+\frac{1}{m^*}\right)}})</td>
<td>-</td>
<td>(\sqrt{\frac{\lambda^s}{\lambda^s+\lambda^f}}\cdot\frac{C_M+C_D}{V^*})</td>
<td>(\sqrt{2(\lambda^s+\lambda^f)C_M+C_D}+\sqrt{2\lambda^sV^*})</td>
</tr>
<tr>
<td>(P_{DV})</td>
<td>(\sqrt{\frac{(m^<em>-1)V^</em>+C_M+C_D}{\lambda^s+\lambda^f}})</td>
<td>-</td>
<td>(2-\frac{2}{r}+\sqrt{\frac{\lambda^s}{\lambda^s+\lambda^f}}\cdot\frac{C_M+C_D}{V^*})</td>
<td>(\sqrt{2(\lambda^s+\lambda^f)\left(V^*+\frac{2-r}{r}V+C_M+C_D\right)})</td>
</tr>
<tr>
<td>(P_{DM})</td>
<td>(\sqrt{\frac{n^<em>(V^</em>+C_M+C_D)}{\lambda^s+\lambda^f}})</td>
<td>(\sqrt{\frac{2\lambda^s}{\lambda^s+\lambda^f}}\cdot\frac{C_M}{V^*+C_M})</td>
<td>-</td>
<td>(2\sqrt{\lambda^s(V^*+C_M)}+\sqrt{2\lambda^fC_D})</td>
</tr>
<tr>
<td>(P_{DM^*})</td>
<td>(\sqrt{\frac{n^*m^<em>V^</em>+n^*C_M+C_D}{\lambda^s+\lambda^f}})</td>
<td>(\sqrt{\frac{\lambda^s}{\lambda^s+\lambda^f}}\cdot\frac{C_M}{V^*})</td>
<td>(\sqrt{\frac{C_M}{V^*}})</td>
<td>(\sqrt{2\lambda^fC_D}+\sqrt{2\lambda^sC_M}+\sqrt{2\lambda^sV^*})</td>
</tr>
<tr>
<td>(P_{DMV})</td>
<td>(\sqrt{\frac{n^<em>(m^</em>-1)V+n^<em>(V^</em>+C_M)+C_D}{\lambda^s+\lambda^f}})</td>
<td>(\sqrt{\frac{\lambda^s}{\lambda^s+\lambda^f}}\cdot\frac{C_D}{V^*-\frac{2-r}{r}V+C_M})</td>
<td>(2-\frac{2}{r})</td>
<td>(\sqrt{2\lambda^fC_D}+\sqrt{2\lambda^s\left(V^*+\frac{2-r}{r}V+C_M\right)})</td>
</tr>
<tr>
<td>(P_{DMV^*})</td>
<td>(\sqrt{\frac{n^<em>(m^</em>-1)V+n^<em>(V^</em>+C_M)+C_D}{\lambda^s+\lambda^f}})</td>
<td>(\sqrt{\frac{\lambda^s}{\lambda^s+\lambda^f}}\cdot\frac{C_D}{V^*-\frac{2-r}{r}V+C_M})</td>
<td>(2-\frac{2}{r})</td>
<td>(\sqrt{2\lambda^fC_D}+\sqrt{2\lambda^s\left(V^*+\frac{2-r}{r}V+C_M\right)})</td>
</tr>
</tbody>
</table>

Algorithmes d’ordonnancement et schémas de résilience pour les pannes et les erreurs silencieuses
Summary and Future Work

Resilience patterns:
- Checkpointing for fail-stop errors
- Verifications for silent errors
- Multi-level checkpointing for both
- Models and optimal solutions

Currently, I work on designing new detection techniques
Algorithm Based Fault Tolerance (ABFT)

Any application specific technique used to cope with faults.

Consider the blocked matrix multiplication $C = A \times B$.

<table>
<thead>
<tr>
<th>Application Workflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>for $i = 1$ to $\lceil \frac{m}{b} \rceil$ do</td>
</tr>
<tr>
<td>for $j = 1$ to $\lceil \frac{m}{b} \rceil$ do</td>
</tr>
<tr>
<td>for $k = 1$ to $\lceil \frac{m}{b} \rceil$ do</td>
</tr>
<tr>
<td>$C_{i,j} \leftarrow C_{i,j} + A_{i,k} \times B_{k,j}$</td>
</tr>
</tbody>
</table>

\( m \) matrix size  
\( b \) block size

ABFT can be used to add per-block verification.
Algorithm Based Fault Tolerance (ABFT)

Any application specific technique used to cope with faults.
Consider the blocked matrix multiplication $C = A \times B$.

Application Workflow

\[
\begin{align*}
\text{for } i & = 1 \text{ to } \left\lceil \frac{m}{b} \right\rceil \text{ do} \\
\quad \text{for } j & = 1 \text{ to } \left\lceil \frac{m}{b} \right\rceil \text{ do} \\
\quad \quad \text{for } k & = 1 \text{ to } \left\lceil \frac{m}{b} \right\rceil \text{ do} \\
\quad \quad \quad C_{i,j} & \leftarrow C_{i,j} + A_{i,k} \times B_{k,j}
\end{align*}
\]

\[m \text{ matrix size} \]
\[b \text{ block size} \]

ABFT can be used to add per-block verification.
Algorithm Based Fault Tolerance

Let $e^T = [1, 1, \cdots, 1]$, we define

$$A^c := \begin{pmatrix} A \\ e^T A \end{pmatrix}, \quad B^r := \begin{pmatrix} B & Be \end{pmatrix}, \quad C^f := \begin{pmatrix} C & Ce \\ e^T C & e^T Ce \end{pmatrix}.$$

Where $A^c$ is the column checksum matrix, $B^r$ is the row checksum matrix and $C^f$ is the full checksum matrix.

$$A^c \times B^r = \begin{pmatrix} A \\ e^T A \end{pmatrix} \times \begin{pmatrix} B & Be \end{pmatrix} = \begin{pmatrix} AB & ABe \\ e^T AB & e^T ABe \end{pmatrix} = \begin{pmatrix} C & Ce \\ e^T C & e^T Ce \end{pmatrix} = C^f.$$
Let $e^T = [1, 1, \ldots, 1]$, we define

$$A^c := \begin{pmatrix} A \\ e^T A \end{pmatrix}, \quad B^r := \begin{pmatrix} B & B e \end{pmatrix}, \quad C^f := \begin{pmatrix} C \\ e^T C \\ e^T C e \end{pmatrix}.$$ 

Where $A^c$ is the **column checksum matrix**, $B^r$ is the **row checksum matrix** and $C^f$ is the **full checksum matrix**.

$$A^c \times B^r = \begin{pmatrix} A \\ e^T A \end{pmatrix} \times \begin{pmatrix} B & B e \end{pmatrix} = \begin{pmatrix} AB & AB e \\ e^T AB & e^T AB e \end{pmatrix} = \begin{pmatrix} C \\ e^T C \\ e^T C e \end{pmatrix} = C^f.$$
ABFT: Detection

Let us build a small example:

\[ A^c = \begin{pmatrix} 1 & 2 & 0 \\ 2 & 1 & 0 \\ 2 & 1 & 2 \\ 5 & 4 & 2 \end{pmatrix}, \quad B^r = \begin{pmatrix} 1 & 1 & 1 & 3 \\ 2 & 0 & 3 & 5 \\ 0 & 2 & 2 & 4 \end{pmatrix}, \]

\[ C^f = A^c \times B^r = \begin{pmatrix} 5 & 1 & 7 & 13 \\ 4 & 3 & 5 & 11 \\ 4 & 6 & 9 & 19 \\ 13 & 9 & 21 & 43 \end{pmatrix} \]

Everything seems fine. However, a silent error has occurred!

Indeed, recomputing the checksums we find that:

\[ \begin{pmatrix} 5 + 1 + 7 = 13 \\ 4 + 3 + 5 = 12 \\ 4 + 6 + 9 = 19 \\ 13 + 10 + 21 = 44 \end{pmatrix} \]

Checksums do not match!
ABFT: Detection

Let us build a small example:

\[ A^c = \begin{pmatrix} 1 & 2 & 0 \\ 2 & 1 & 0 \\ 2 & 1 & 2 \\ 5 & 4 & 2 \end{pmatrix}, \quad B^r = \begin{pmatrix} 1 & 1 & 1 & 3 \\ 2 & 0 & 3 & 5 \\ 0 & 2 & 2 & 4 \end{pmatrix}, \]

\[ C^f = A^c \times B^r = \begin{pmatrix} 5 & 1 & 7 & 13 \\ 4 & 3 & 5 & 11 \\ 4 & 6 & 9 & 19 \\ 13 & 9 & 21 & 43 \end{pmatrix} \]

Everything seems fine. However, a silent error has occurred!

Indeed, recomputing the checksums we find that:

\[ \begin{pmatrix} 5 + 1 + 7 = 13 \\ 4 + 3 + 5 = 12 \\ 4 + 6 + 9 = 19 \\ 13 + 10 + 21 = 44 \end{pmatrix} \]

Checksums do not match!
ABFT: Detection

Let us build a small example:

\[ A^c = \begin{pmatrix}
1 & 2 & 0 \\
2 & 1 & 0 \\
2 & 1 & 2 \\
5 & 4 & 2
\end{pmatrix}, \quad B^r = \begin{pmatrix}
1 & 1 & 1 & 3 \\
2 & 0 & 3 & 5 \\
0 & 2 & 2 & 4
\end{pmatrix}, \]

\[ C^f = A^c \times B^r = \begin{pmatrix}
5 & 1 & 7 & 13 \\
4 & 3 & 5 & 11 \\
4 & 6 & 9 & 19 \\
13 & 9 & 21 & 43
\end{pmatrix} \]

Everything seems fine. However, a silent error has occurred!

Indeed, recomputing the checksums we find that:

\[ \begin{pmatrix}
5 + 1 + 7 = 13 \\
4 + 3 + 5 = 12 \\
4 + 6 + 9 = 19 \\
13 + 10 + 21 = 44
\end{pmatrix} \]

Checksums do not match!
ABFT: Detection

Let us build a small example:

\[
A^c = \begin{pmatrix} 1 & 2 & 0 \\ 2 & 1 & 0 \\ 2 & 1 & 2 \\ 5 & 4 & 2 \end{pmatrix}, \quad B^r = \begin{pmatrix} 1 & 1 & 1 & 3 \\ 2 & 0 & 3 & 5 \\ 0 & 2 & 2 & 4 \end{pmatrix},
\]

\[
C^f = A^c \times B^r = \begin{pmatrix} 5 & 1 & 7 & 13 \\ 4 & 3 & 5 & 11 \\ 4 & 6 & 9 & 19 \\ 13 & 9 & 21 & 43 \end{pmatrix}
\]

Everything seems fine. However, a silent error has occurred!

Indeed, recomputing the checksums we find that:

\[
\begin{pmatrix} 5 + 1 + 7 = 13 \\ 4 + 3 + 5 = 12 \\ 4 + 6 + 9 = 19 \\ 13 + 10 + 21 = 44 \end{pmatrix}
\]

Checksums do not match!
ABFT: Correction

\[ C^f = A^c \times B^r = \begin{pmatrix} 5 & 1 & 7 & 13 \\ 4 & 3 & 5 & 11 \\ 4 & 6 & 9 & 19 \\ 13 & 9 & 21 & 43 \end{pmatrix} \begin{pmatrix} 5 + 1 + 7 = 13 \\ 4 + 3 + 5 = 12 \\ 4 + 6 + 9 = 19 \\ 13 + 10 + 21 = 44 \end{pmatrix} \]

Both checksums are affected, giving out the location of the error.

We solve:

\[
\begin{align*}
4 + x + 5 &= 11 \\
x &= 11 - 5 - 4 = 2
\end{align*}
\]

\[
\begin{align*}
1 + x + 6 &= 9 \\
x &= 9 - 6 - 1 = 2
\end{align*}
\]

Recomputing the checksums we find that:

\[
\begin{pmatrix} 5 + 1 + 7 = 13 \\ 4 + 2 + 5 = 11 \\ 4 + 6 + 9 = 19 \\ 13 + 9 + 21 = 43 \end{pmatrix}
\]

Checksums match!
ABFT: Correction

\[ C^f = A^c \times B^r = \begin{pmatrix} 5 & 1 & 7 & 13 \\ 4 & 3 & 5 & 11 \\ 4 & 6 & 9 & 19 \\ 13 & 9 & 21 & 43 \end{pmatrix}, \begin{pmatrix} 4 + 3 + 5 = 12 \\ 4 + 6 + 9 = 19 \\ 13 + 10 + 21 = 44 \end{pmatrix} \]

Both checksums are affected, giving out the location of the error.

We solve:

\[
\begin{align*}
4 + x + 5 &= 11 \\
x &= 11 - 5 - 4 = 2
\end{align*}
\]

\[
\begin{align*}
1 + x + 6 &= 9 \\
x &= 9 - 6 - 1 = 2
\end{align*}
\]

Recomputing the checksums we find that:

\[
\begin{pmatrix} 5 + 1 + 7 = 13 \\ 4 + 2 + 5 = 11 \\ 4 + 6 + 9 = 19 \\ 13 + 9 + 21 = 43 \end{pmatrix}
\]

Checksums match!
ABFT: Correction

\[ C^f = A^c \times B^r = \begin{pmatrix} 5 & 1 & 7 & 13 \\ 4 & 3 & 5 & 11 \\ 4 & 6 & 9 & 19 \\ 13 & 9 & 21 & 43 \end{pmatrix}, \begin{pmatrix} 5 + 1 + 7 = 13 \\ 4 + 3 + 5 = 12 \\ 4 + 6 + 9 = 19 \\ 13 + 10 + 21 = 44 \end{pmatrix} \]

Both checksums are affected, giving out the location of the error.

We solve:

\[
4 + x + 5 = 11 \quad 1 + x + 6 = 9 \\
x = 11 - 5 - 4 = 2 \quad x = 9 - 6 - 1 = 2
\]

Recomputing the checksums we find that:

\[
\begin{pmatrix} 5 + 1 + 7 = 13 \\ 4 + 2 + 5 = 11 \\ 4 + 6 + 9 = 19 \\ 13 + 9 + 21 = 43 \end{pmatrix} \quad \text{Checksums match!}
\]