Black or White? How to Develop an AutoTuner for Memory-based Analytics

Mayuresh Kunjir and Shivnath Babu

Review by Ross Tooley
Background
Apache Spark

Bulk-synchronous parallel

Distributed on cluster
The optimisation problem

Input

- Program
- Data

Optimiser

Output

System-configuration parameters

- Containers per node
- Task concurrency
- Cache capacity
- Shuffle capacity
- New ratio
Existing Spark optimisers

- Guidelines for manual tuning
  - Spark
  - Fekry et al.
- Pure BO
  - Yu et al.
- Genetic algorithms
  - Wang et al.
- Regression trees
- And more...
Paper overview

Performance model  
RelM  
BO & RL  
Evaluation
RelM
Design

RelM

Application Profile → Statistics Generator →_INITIALIZER → ArbTrator → Selector

Config \( c \) and \( U_c \)

Reliable \( C \)

Container Sizes

Recommendation \( C^* \)
Components

Statistics

| Notation | Description | Example |
|----------|-------------|---------|
| N        | Containers per Node | 1       |
| M_h      | Heap size    | 4404MB  |
| CPU_{avg} | Average CPU usage | 35%     |
| Disk_{avg} | Average disk usage | 2%      |
| M_i      | Code Overhead 90%ile value | 115MB    |
| M_c      | Cache Storage 90%ile value | 2500MB   |
| M_r      | Task Shuffle 90%ile value | 0MB      |
| M_u      | Task Unmanaged 90%ile value | 770MB    |
| P        | Task Concurrency | 2       |
| H        | Cache Hit Ratio (the fraction of cached data partitions actually read from cache) | 0.3     |
| S        | Data Spillage Fraction (the fraction of shuffle data spilled to disk) | 0       |

Enumerator

| # containers | Task concurrency |
|--------------|------------------|
| (1,1)        | (2,1)            |
| (1,2)        | (3,1)            |
| ...          |                  |

Initialiser

\[
\begin{align*}
  m_c &= m_h \cdot \min \left( \frac{M_c}{H + M_h}, 1 - \delta \right) \\
  m_s &= \min \left( \frac{M_i}{1 - S/P}, (1 - \delta) \cdot m_h \right) \\
  m_a &= \min \left( \frac{NR}{NR + 1} \cdot m_c + m_h \cdot \frac{1}{NR + 1} \cdot SR \cdot 2 \right) \\
  m_i &= \min \left( \frac{1 - (1 - \delta) \cdot 100}{n \cdot CPU_{avg}/P}, p^{\text{risk}} \right) \\
  p^{\text{CPU}} &= \frac{1 - (1 - \delta) \cdot 100}{n \cdot CPU_{avg}/P} \\
  p^{\text{Disk}} &= \frac{1 - (1 - \delta) \cdot 100}{n \cdot Disk_{avg}/P} \\
  \rho &= \min \left( \frac{1 - (1 - \delta) \cdot m_h}{M}, p = \min(p^{\text{CPU}}, p^{\text{risk}}, p^{\text{Disk}}) \right)
\end{align*}
\]

Arbitrator

Algorithm 1 ReM Arbitrator

Input: Configuration \( e = (M_h, M_c, p, m_c, m_h) \), Safety factor \( \delta \)

1. if \( (M_i + M_c) > (1 - \delta) \cdot m_h \) then
2. return flagging insufficient memory
3. end if
4. while \( (M_i + p \cdot M_h + m_i) > m_h \) do
5. one of the following three in a round-robin manner:
6. I. Decrease \( p \) by 1 if \( p > 1 \)
7. II. Reduce \( m_i \) by \( M_h \) ensuring that \( m_i > 0 \)
8. III. Increase \( m_h \) by \( M_h \) ensuring \( m_h < (1 - \delta) \cdot m_h \)
9. end while
10. Set shuffle memory \( m_i = \min(m_i, 0.5 \cdot m_i/p) \)
11. Set output \( C = (M_h, M_c, p, m_c, m_h) \)

Selector

\[
U_C = \frac{M_i + m_c + p^* (M_h + m_h)}{m_i}
\]
Bayesian Optimisation and Reinforcement Learning
What’s the idea here?

1. Quasi-parameters

\[ q_1^x = \frac{M_i + \min(m_c^x, m_c) + p^x \cdot (M_u + \min(m_s^x, m_s))}{m_h^x} \]

\[ q_2^x = \frac{M_i + m_c}{\min(m_0^x, m_c^x)} \]

\[ q_3^x = \frac{p^x \cdot \min(m_s^x, m_s)}{0.5 \cdot m_e^x} \]

\[ q^x = \{q_1^x, q_2^x, q_3^x\} \]

2. Extend input space with \( q \)

\[ \{x_1, x_2, x_3, x_4, x_5, q_1^x, q_2^x, q_3^x\} \]

3. Use RelM to predict quasi-parameters for each real configuration
Bayesian Optimisation

\[ GP(x, y) \]

\[ GP(x \cup q, y) \]
Comparison to BOAT

\[ GP(x \cup q, y) \]

\[ GP(x, q_1) \quad GP(x, q_2) \quad GP(x, q_3) \]

\[ GP(q, y) \]
Reinforcement Learning (DDPG)
Results
Performance improvement

Figure 15: Runtime of every recommended configuration is scaled to the runtime of MaxResourceAllocation. Number of failed containers is shown on top of bars.
Convergence time

Figure 14: Training overheads of tuning policies. Number of iterations is shown on top of bars.
Looks good... any issues?

Overfitting
Comparisons
Comparing all optimisers

The general structure

Input → Modelling → Searching → Output
Contrasting search methods

- **Bayesian Optimisation**
  - BOAT
  - Kunjir & Babu
  - CherryPick

- **Reinforcement Learning**
  - DDPG
    - QTune
    - Kunjir & Babu
  - REINFORCE
    - Mirhoseini et al.
  - Thompson Sampling
    - Bao

- **Genetic**
  - REGAL

- **Population-Based Training**
  - Jaderberg et al.

- **Back-tracking**
  - TASO
Contrasting model methods

Grey Box

Expert model
- BOAT
- TASO
- QTune
- Kunjir & Babu

Pure search-based
- CherryPick
- Bao
- Jaderberg et al

Auto-encoding
- REGAL (GNN)
- Mirhoseni et al (FFN)

Black Box