Brief Overview of Knobs Tuning in Self-Tuning Databases

Hong Wu
Agenda

• Knobs Tuning overview
• Deep Dive into BestConfig
• Deep Dive into iTuned, Ottertune
  • Brief introduction of Bayesian Optimization + Gaussian Processing
• Deep Dive into cdbtune
• Demo
• Comparison & Discussion
Self-Tuning Databases

• Physical Design Advice
  • Optimal Indexes
  • Materialized Views
  • Partitioning Schemes

• **Knobs/Configuration Tuning**

Knobs Tuning for Systems

• General System Tuning
  • BestConfig
  • Hill-Climbing

• Specific System Tuning
  • Hadoop (Big Data)
    • Starfish, Aloja-ml
  • Database
    • Ottertune, cdbtune, iTuned
  • Microprocessor
    • COMT
Knobs Tuning for Database

• Knowledge/Rules Based (Specific Parameter Tuning)
  • DB2 (also use control theory)
  • SQL Server
  • Oracle

• Control Theory (define a Mathematical Model then solve it)
  • COMFORT

• Search Based (How to sample effectively)
  • BestConfig
  • SARD

• Model Based
  • Bayesian Optimization (Ottertune, iTuned)
  • Reinforcement Learning (cdbtune)
  • Semi-supervised (COMT)
Knobs Tuning for Database

• [1]. Duan, Songyun, et al. "Tuning database configuration parameters with iTuned." (VLDB 09)
BestConfig

SUT: System Under Tune
PO: Performance Optimization
Subproblem1: Sampling

• Three requirements/challenges
  • the set has a wide coverage over the high-dimensional space of configuration parameters
  • the set is small enough to meet the resource limit and to reduce test costs
  • the set can be scaled to have a wider coverage, if the resource limit is expanded
Subproblem 2: Performance Optimization

Maximize the performance metric based on the given # samples

Three requirements/Challenges

- it can find an answer even with a limited set of samples;
- it can find a better answer if a larger set of samples is provided;
- it will not be stuck in local sub-optimal areas and has the possibility to find the global optimum, given enough resources.
Figure 1: Diverging performance surfaces of MySQL, Tomcat and Spark. (Best view in color)
Solutions

• Divide-and-Diverge Sampling (DDS) method
  • Impact of an influential parameter’s values on the performance can be demonstrated through comparisons of performances, disregard of other parameters’ values.

• Recursive Bound-and-Search (RBS) algorithm
  • Given a continuous surface, there is a high possibility that we find other points with similar or better performances around the point with the best performance in the sample set.
Tuning Algorithm

• Recursively go from the first step through the third step till the resource limit is reached
  • 1. Search the best setting in the sample set
    • if not exist, go to step 3 for the whole space
  • 2. Bound the unexplored subspace around the best setting
  • 3. Invoke the sampling method to get a new sample set

• Given a limit of nr tests, RBS can run in r rounds with each sample set sized n
BestConfig Advantages

• Can apply to general systems
  • Spark, Hadoop, Hive, Cassandra, MySQL, Tomcat
• Only limited samples required
• Flexible plugin tuning framework
• Scalable comparing to ML methods
  • more samples, more accurate
Potential Issues

• Difference between DDS and LHS?
• Results of the Model Based experiments are doubtful
iTuned
Bayesian Optimization for Tuning Knobs

• Frequentist Inference: $p(X|\Theta)$

• Bayesian Inference: $p(\Theta|X) = \frac{p(X|\Theta) \cdot p(\Theta)}{p(X)}$
  
  • $\Theta$: Cause X: results
  • $p(\Theta)$: prior belief prob
  • $p(\Theta|X)$: posterior prob, $p(X|\Theta)$: likelihood

• Feedback-driven experiments
  • Pick some priors the knobs will behave
  • Search the knobs space, update the prior belief based on experiments
    • Exploration vs Exploitation
Bayesian Optimization for Tuning Knobs

1. Using previous evaluated points $X$s computer a posterior expectation of the loss (prior function or distribution, e.g. Gaussian Process)

2. Sample the loss at a new point that maximize some utility of the expectation
   - The utility tells us where is the best place to sample future points

$$IP(X) = \begin{cases} y(X^*) - y(X) & \text{if } y(X) < y(X^*) \\ 0 & \text{otherwise} \end{cases}$$

$$EIP(X) = \max_{x \in \text{DOM}} EIP(X)$$

$$X_{\text{next}} = \text{arg} \max_{x \in \text{DOM}} EIP(X)$$

$$EIP(X) = \int_{p=-\infty}^{p=+\infty} IP(X) \text{pdf} \hat{y}(x)(p) dp$$

$$EIP(X) = \int_{p=-\infty}^{p=y(X^*)} (y(X^*) - p) \text{pdf} \hat{y}(x)(p) dp$$

$$EIP(X) = \int_{p=y(X^*)}^{p=+\infty} IP(X) \text{pdf} \hat{y}(x)(p) dp$$
Gaussian Processing – The Idea

Drawing samples from the prior

10 samples from the prior distribution using a squared exponential kernel (Eq. 1) with \( l = 0.2 \) and \( \sigma = 2.5 \). The dark grey line indicates \( m(x) \), the gray area indicates the 95% confidence region, i.e. \( m(x) \pm 2\sqrt{K(x)} = m(x) \pm 2\sigma \).
Gaussian Processing - Hyperparameter

Mean: also 0,
Covariance matrix:

\[ k(x, x') = \sigma_f^2 \exp \left[ -\frac{(x - x')^2}{2l^2} \right] + \sigma_n^2 \delta(x, x'), \]

\( l \): influence between data points
\( \sigma \): how far away deviate from mean
Gaussian Processing Summary

• A non-parametric model
• For example, given \{X_i, Y_i\} which is evaluated points, given a unevaluated point \(x^*\), Gaussian Processing can regress a \(y\) value within a distribution
iTuned Optimization

Adaptive Sampling: Algorithm run by iTuned’s Planner
1. Initialization: Conduct experiments based on Latin Hypercube Sampling, and initialize GRS and $X^* = \arg \min_{i} y(X^{(i)})$ with collected samples;
2. Until the stopping condition is reached, do
3. Find $X_{next} = \arg \max_{X \in DOM} EIP(X)$;
4. Executor conducts the next experiment at $X_{next}$ to get a new sample;
5. Update the GRS and $X^*$ with the new sample; Go to Line 2;

Figure 2: Steps in iTuned’s Adaptive Sampling algorithm

$L^* = \arg \max \min \text{dist}(X^{(j)}, X^{(k)})$

\begin{align*}
\text{pdf } \tilde{y}(X)(p) &= \frac{1}{\sqrt{2\pi v(X)}} \exp\left(-\frac{(p - u(X))^2}{2v^2(X)}\right) \\
EIP(X) &= v(X)(\mu(X)\Phi(\mu(X)) + \phi(\mu(X)))
\end{align*}

Here, $\mu(X) = \frac{y(X^*) - u(X)}{v(X^*)}$. $\Phi$ and $\phi$ are $N(0, 1)$ Gaussian
iTuned Optimization

Figure 4: GRS from five samples (from Example 1)

Figure 5: Example of EIP computation (from Example 2)
Potential Issues

• Cold Start
• Cost to calculate EIP
• How to choose hyperparameter of GP(maximum likelihood estimation?)
• In Recommendation Phase, use Gradient Descent instead of sampling
• Prior of GP: x1 and x2 are close, f(x1) and f(x2) should be similar.
OtterTune

Figure 3: OtterTune Machine Learning Pipeline – This diagram shows the processing path of data in OtterTune. All previous observations reside in its repository. This data is first then passed into the **Workload Characterization** (Sect. 4) component that identifies the most distinguishing DBMS metrics. Next, the **Knob Identification** (Sect. 5) component generates a ranked list of the most important knobs. All of this information then fed into the **Automatic Tuner** (Sect. 6) component where it maps the target DBMS’s workload to a previously seen workload and generates better configurations.
Ottertune

- Inference pipeline of ottertune
  - Workload Characterization
  - Knobs Identification
  - Recommendation

- Training:
  - Knobs Tuning
Ottertune – Phase 1

• Workload Characterization
  • Use FA (Factor Analysis) to reduce dimensions of the metrics (internal + external)
  • Use Kmeans to cluster typical workloads
Ottertune – Phase2

• Knobs Identification
  • Lasso
Ottertune – Phase3

• Recommendation
Ottertune – The Model

• Knobs Tuning
  • Gaussian Processing Regression + Gradient Descent
Potential Issues

• Not end-to-end model, phases relate to each other
  • E.g. if #knobs after lasso is large, not sure about the model performance
  • Can it learn knobs hierarchy?

• Highly depend on training data(data repository), can not transfer between instances

• Gaussian Processing has issues in high-dimensional continous space?

• Gradient Descent might go to local optimal
cdbtune

• End to end using Reinforcement Learning
  • DDPG method to handle continuous space action
Potential Issues

• Training (30 nodes using 4.7 hours, 2000 iterators)
  • How to train RL in parallel? (model parallelize?)
  • Only 2000 iterators can converge?
Ottertune Demo
Comparison & Discussion
• Duan, Songyun, et al. "Tuning database configuration parameters with iTuned." (VLDB 09)
• Van Aken, Dana, et al. “Automatic database management system tuning through large-scale machine learning." (SIGMOD 17)
• Zhu, Yuqing, et al. "Bestconfig: tapping the performance potential of systems via automatic configuration tuning." (SOCC 17)
• Xi, Bowei, et al. "A smart hill-climbing algorithm for application server configuration." (WWW 04)
• Guo, Qi, et al. "Effective and efficient microprocessor design space exploration using unlabeled design configurations.”
• Sullivan, David G., Margo I. Seltzer, and Avi Pfeffer. Using probabilistic reasoning to automate software tuning.
Thank You