BOAT: Building Auto-Tuners with Structured Bayesian Optimization

BespOke Auto-Tuners

Indigo Orton – R244
Key idea

- Bespoke auto-tuners for systems
- Inject developer insight
- Faster tuning
• Diversity of workloads – one size doesn’t fit all
• Configuration tuning – non-trivial
• Optimal configuration – moving target
Motivation – Auto-Tuners

• Auto-tuners – tuning is not always intuitive
• Many iterations
• Costly performance evaluation, generic tuners, & you
• High dimensionality
Example – Cassandra

- Built for high throughput
- JVM based – garbage collection pauses
- Tuning garbage collection
- 99th percentile – 19ms -> 7ms

Figure from BOAT [1]
Example – Cassandra

- BOAT – within 10% of best after 2 iterations
- Spearmint [2] - 16 iterations, 4 hours

Figure from BOAT [1]
Bayesian Optimization

- Basis of most generic auto-tuners
- Probabilistic modeling of objective function
- Gaussian process
- Curse of dimensionality – too many iterations
Structured Bayesian Optimization

- Extension of Bayesian Optimization
- Gaussian process -> developer structured probabilistic model
- Insight into objective function - incrementally
- Happy medium
Incremental structure

- Models Eden size
- Tuner models and minimizes latency
- Larger search space

```
struct GCRateModel : public SemiParametricModel<GCRateModel> {
    GCRateModel() {
        allocated_mbs_per_sec =
            std::uniform_real_distribution<>(0.0, 5000.0)(generator);
        // Omitted: also sample the GP parameters
    }

double parametric(double eden_size) const {
    // Model the rate as inversely proportional to Eden's size
    return allocated_mbs_per_sec / eden_size;
}

double allocated_mbs_per_sec;
};
```

- Models latency
- Tuner minimizes set model of latency
- Smaller search space

```
struct CassandraModel : public DAGModel<CassandraModel> {
    void model(int ygs, int sr, int mtt){
        // Calculate the size of the heap regions
        double es = ygs * sr / (sr + 2.0); // Eden space's size
        double ss = ygs / (sr + 2.0); // Survivor space's size
        // Define the dataflow between semi-parametric models
        double rate = output("rate", rate_model, es);
        double duration = output("duration", duration_model, es, ss, mtt);
        double latency = output("latency", latency_model, rate, duration, es, ss, mtt);
    }
}
```

Figure from BOAT [1]
Results – NN Training

• High dimensionality
• Optimize NN training
• Optimal distribution architecture based on available machines
• Communication time calculation (a max function) – hard to auto fit, easy to manually model
• 2 hour tuning time – large net benefit
Results – NN Training

Figure from BOAT [1]
Context

- OpenTuner [3] – domain specific search techniques
- Spearmint [2] – traditional Bayesian Optimization
Encouraging highlights

• Practical integration of developer knowledge
• Retains benefits of auto-tuners
• Handles high dimensionality
Further questions

- Tuning the tuner
- Incremental structure – is there a heuristic?
- Model of parameters – configuration chooser
Conclusion

• Auto-tuning

• Inject developer insight

• Structured Bayesian Optimization

• Curse of dimensionality

• Happy medium

