# Today's Agenda

- 08:30 Welcome and broader context (Saman Amarasinghe)
- **08:40 Introduction to OpenTuner (Jason Ansel)**
- 09:10 Search techniques (Kalyan Veeramachaneni)
- 09:35 In depth example (Jeffrey Bosboom)
- 10:00 Break
- 10:15 Applications
  - Halide (Jonathan Ragan-Kelley)
  - SEJITS (Chick Markley)
  - JVM optimization (Tharindu Rusira)
- 11:00 Hands on session (Shoaib Kamil)
- 11:45 Discussion

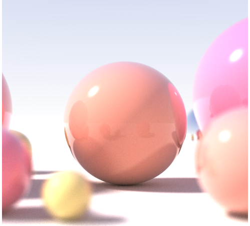
# Introduction to OpenTuner


Jason Ansel

MIT - CSAIL

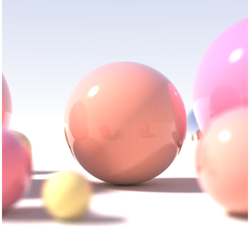

Febuary 8, 2015

# Raytracer Example



An example ray tracer program: `raytracer.cpp`

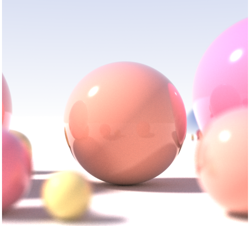
# Raytracer Example



An example ray tracer program: `raytracer.cpp`

```
$ g++ -O3 -o raytracer_a raytracer.cpp
$ time ./raytracer_a
./raytracer_a  0.17s user 0.00s system 99% cpu 0.175 total
```

# Raytracer Example



An example ray tracer program: `raytracer.cpp`

```
$ g++ -O3 -o raytracer_a raytracer.cpp
$ time ./raytracer_a
./raytracer_a  0.17s user 0.00s system 99% cpu 0.175 total
```

1.47x speedup with:

```
$ g++ -O3 -o raytracer_b apps/raytracer.cpp -funsafe-math-optimizations -fwrapv
    ↪ -fno-expensive-optimizations --param=max-peel-branches=115 -fweb -fno-
    ↪ cx-fortran-rules  --param=max-inline-recursive-depth=25 -fno-btr-bb-
    ↪ exclusive -fno-tree-ch --param=iv-max-considered-uses=69 -fgcse-las -
    ↪ ftree-loop-distribution --param=max-goto-duplication-insns=11 --param=
    ↪ max-hoist-depth=44 -fsched-stalled-insns-dep --param=max-once-peeled-
    ↪ insns=165 --param=max-pipeline-region-insns=316 --param=iv-consider-all
    ↪ -candidates-bound=75
$ time ./raytracer_b
./raytracer_b  0.12s user 0.00s system 99% cpu 0.119 total
```

# iv-consider-all-candidates-bound what???

This command is brittle and confusing:

```
$ g++ -O3 -o raytracer_b apps/raytracer.cpp -funsafe-math-optimizations -fwrapv
    ↪ -fno-expensive-optimizations --param=max-peel-branches=115 -fweb -fno-
    ↪ cx-fortran-rules  --param=max-inline-recursive-depth=25 -fno-btr-bb-
    ↪ exclusive -fno-tree-ch --param=iv-max-considered-uses=69 -fgcse-las -
    ↪ ftree-loop-distribution --param=max-goto-duplication-insns=11 --param=
    ↪ max-hoist-depth=44 -fsched-stalled-insns-dep --param=max-once-peeled-
    ↪ insns=165 --param=max-pipeline-region-insns=316 --param=iv-consider-all
    ↪ -candidates-bound=75
```

# iv-consider-all-candidates-bound what???

This command is brittle and confusing:

```
$ g++ -O3 -o raytracer_b apps/raytracer.cpp -funsafe-math-optimizations -fwrapv
    ↪ -fno-expensive-optimizations --param=max-peel-branches=115 -fweb -fno-
    ↪ cx-fortran-rules --param=max-inline-recursive-depth=25 -fno-btr-bb-
    ↪ exclusive -fno-tree-ch --param=iv-max-considered-uses=69 -fgcse-las -
    ↪ ftree-loop-distribution --param=max-goto-duplication-insns=11 --param=
    ↪ max-hoist-depth=44 -fsched-stalled-insns-dep --param=max-once-peeled-
    ↪ insns=165 --param=max-pipeline-region-insns=316 --param=iv-consider-all
    ↪ -candidates-bound=75
```

- Specific to:
  - `raytracer.cpp`
    - Same flags are 1.42*x* **slower** than `-O1` for `fft.c`
  - `GCC 4.8.2-19ubuntu1`
  - `Intel Core i7-4770S`

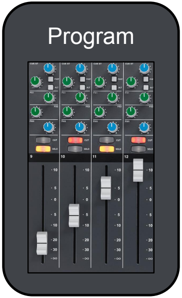
# iv-consider-all-candidates-bound what???

This command is brittle and confusing:

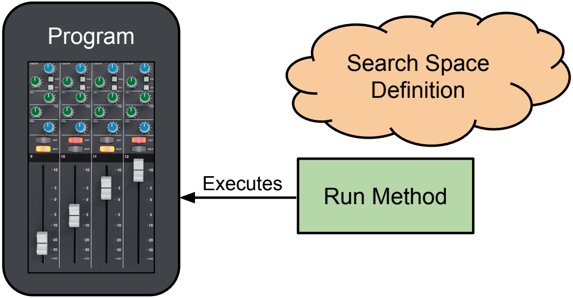
```
$ g++ -O3 -o raytracer_b apps/raytracer.cpp -funsafe-math-optimizations -fwrapv
    ↪ -fno-expensive-optimizations --param=max-peel-branches=115 -fweb -fno-
    ↪ cx-fortran-rules --param=max-inline-recursive-depth=25 -fno-btr-bb-
    ↪ exclusive -fno-tree-ch --param=iv-max-considered-uses=69 -fgcse-las -
    ↪ ftree-loop-distribution --param=max-goto-duplication-insns=11 --param=
    ↪ max-hoist-depth=44 -fsched-stalled-insns-dep --param=max-once-peeled-
    ↪ insns=165 --param=max-pipeline-region-insns=316 --param=iv-consider-all
    ↪ -candidates-bound=75
```

- Specific to:
  - `raytracer.cpp`
    - Same flags are 1.42$x$ **slower** than `-O1` for `fft.c`
  - `GCC 4.8.2-19ubuntu1`
  - `Intel Core i7-4770S`
- Autotuners can help!

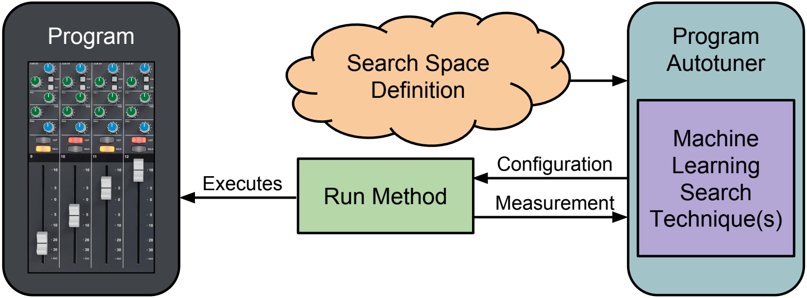
# How to Autotune a Program



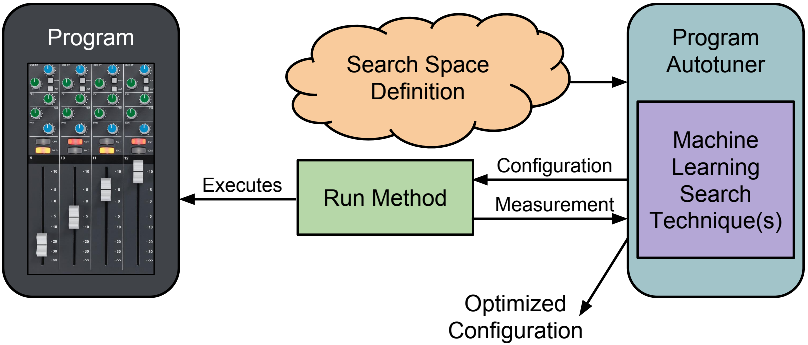
# How to Autotune a Program



# How to Autotune a Program



# How to Autotune a Program



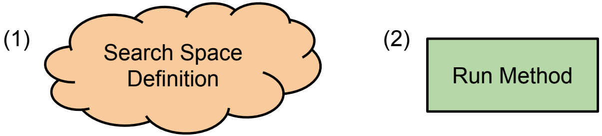
# OpenTuner

- OpenTuner is an general framework for program autotuning
  - Extensible configuration representation
  - Uses ensembles of techniques to provide robustness to different search spaces

# OpenTuner

- OpenTuner is an general framework for program autotuning
  - Extensible configuration representation
  - Uses ensembles of techniques to provide robustness to different search spaces

- As an example, lets implement a GCC flags autotuner with OpenTuner



# Define the Search Space with OpenTuner

- Optimization level: `O0`, `O1`, `O2`, `O3`

```
manipulator = ConfigurationManipulator()
manipulator.add_parameter(IntegerParameter('opt_level', 0, 3))
```

# Define the Search Space with OpenTuner

- Optimization level: `O0`, `O1`, `O2`, `O3`

```
manipulator = ConfigurationManipulator()
manipulator.add_parameter(IntegerParameter('opt_level', 0, 3))
```

- On/off flags, eg: `'-falign-functions'` vs `'-fno-align-functions'`

```
GCC_FLAGS = [
  'align-functions', 'align-jumps', 'align-labels',
  'branch-count-reg', 'branch-probabilities',
  # ... (176 total)
]
for flag in GCC_FLAGS:
  manipulator.add_parameter(EnumParameter(flag, ['on', 'off', 'default']))
```

# Define the Search Space with OpenTuner

- Optimization level: `O0, O1, O2, O3`

```
manipulator = ConfigurationManipulator()
manipulator.add_parameter(IntegerParameter('opt_level', 0, 3))
```

- On/off flags, eg: `'-falign-functions'` vs `'-fno-align-functions'`

```
GCC_FLAGS = [
  'align-functions', 'align-jumps', 'align-labels',
  'branch-count-reg', 'branch-probabilities',
  # ... (176 total)
]
for flag in GCC_FLAGS:
  manipulator.add_parameter(EnumParameter(flag, ['on', 'off', 'default']))
```

- Parameters, eg: `'--param early-inlining-insns=512'`

```
# (name, min, max)
GCC_PARAMS = [
  ('early-inlining-insns', 0, 1000),
  ('gcse-cost-distance-ratio', 0, 100),
  # ... (145 total)
]
for param, min_val, max_val in GCC_PARAMS:
  manipulator.add_parameter(IntegerParameter(param, min_val, max_val))
```

# Defining the Run Function

- Optimization level: `O0`, `O1`, `O2`, `O3`

```python
def run(self, desired_result, program_input, limit):
  cfg = desired_result.configuration.data
  gcc_cmd = 'g++ raytracer.cpp -o ./tmp.bin'
  gcc_cmd += ' -O{0}'.format(cfg['opt_level'])
```

# Defining the Run Function

- Optimization level: `O0`, `O1`, `O2`, `O3`

```python
def run(self, desired_result, program_input, limit):
  cfg = desired_result.configuration.data
  gcc_cmd = 'g++ raytracer.cpp -o ./tmp.bin'
  gcc_cmd += ' -O{0}'.format(cfg['opt_level'])
```

- On/off flags:

```python
  for flag in GCC_FLAGS:
    if cfg[flag] == 'on':
      gcc_cmd += ' -f{0}'.format(flag)
    elif cfg[flag] == 'off':
      gcc_cmd += ' -fno-{0}'.format(flag)
```

- Parameters:

```python
  for param, min_value, max_value in GCC_PARAMS:
    gcc_cmd += ' --param {0}={1}'.format(param, cfg[param])
```

# Defining the Run Function

- Optimization level: `O0`, `O1`, `O2`, `O3`

```
def run(self, desired_result, program_input, limit):
  cfg = desired_result.configuration.data
  gcc_cmd = 'g++ raytracer.cpp -o ./tmp.bin'
  gcc_cmd += ' -O{0}'.format(cfg['opt_level'])
```

- On/off flags:

```
  for flag in GCC_FLAGS:
    if cfg[flag] == 'on':
      gcc_cmd += ' -f{0}'.format(flag)
    elif cfg[flag] == 'off':
      gcc_cmd += ' -fno-{0}'.format(flag)
```

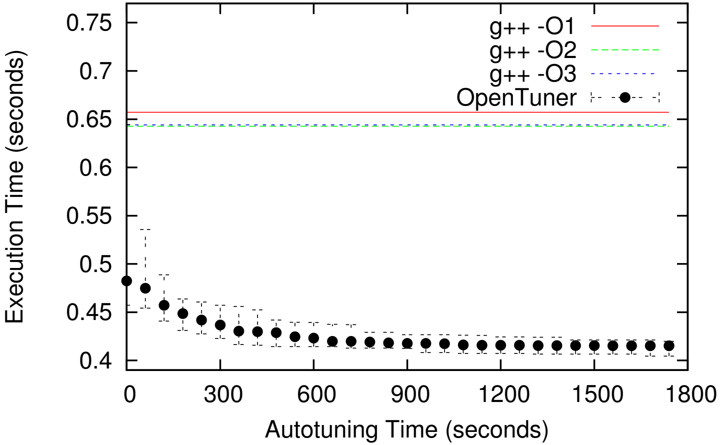
- Parameters:

```
  for param, min_value, max_value in GCC_PARAMS:
    gcc_cmd += ' --param {0}={1}'.format(param, cfg[param])
```

- Measure how well it performs:

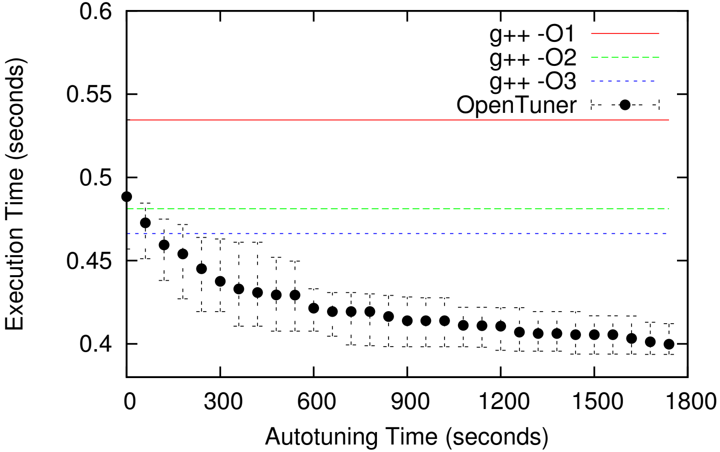
```
  compile_result = self.call_program(gcc_cmd)
  run_result = self.call_program('./tmp.bin')
  return Result(time=run_result['time'])
```

# OpenTuner Results for GCC Flags



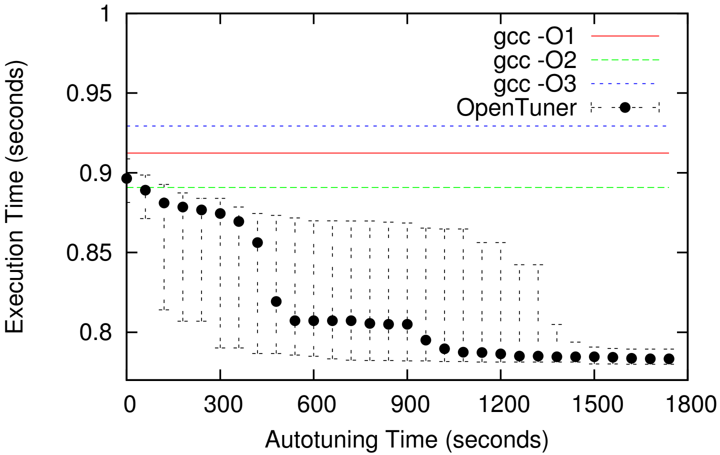

Autotune GCC flags for Ray Tracer. Median of 30 runs, error bars are 20th and 80th percentiles.

# OpenTuner Results for GCC Flags



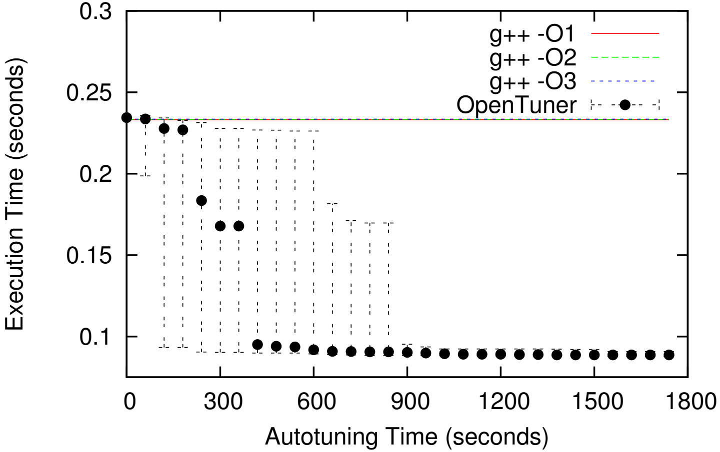

Autotune GCC flags for TSP GA. Median of 30 runs, error bars are 20th and 80th percentiles.

# OpenTuner Results for GCC Flags




Autotune GCC flags for FFT. Median of 30 runs, error bars are 20th and 80th percentiles.

# OpenTuner Results for GCC Flags




Autotune GCC flags for Matrix Multiply. Median of 30 runs, error bars are 20th and 80th percentiles.

# Related Projects

A small selection of many related projects:

| Package | Domain | Search Method |
|---|---|---|
| ATLAS | Dense Linear Algebra | Exhaustive |
| Code Perforation | Compiler | Exhaustive + Simulated Annealing |
| FFTW | Fast Fourier Transform | Exhaustive / Dynamic Prog. |
| OSKI | Sparse Linear Algebra | Exhaustive + Heuristic |
| Periscope | HPC | Exhaustive + Nelder-Mead |
| Active Harmony | Runtime System | Nelder-Mead |
| PATUS | Stencil Computations | Nelder-Mead or Evolutionary |
| Sepya | Stencil Computations | Random-Restart Gradient Ascent |
| Dynamic Knobs | Runtime System | Control Theory |
| Milepost GCC / cTuning | Compiler | IID Model + Central DB |
| SEEC / Heartbeats | Runtime System | Control Theory |
| Insieme | Compiler | Differential Evolution |
| PetaBricks | Programming Language | Bottom-up Evolutionary |
| SPIRAL | DSP Algorithms | Pareto Active Learning |

# Related Projects

A small selection of many related projects:

| Package | Domain | Search Method |
|---|---|---|
| ATLAS | Dense Linear Algebra | Exhaustive |
| Code Perforation | Compiler | Exhaustive + Simulated Annealing |
| FFTW | Fast Fourier Transform | Exhaustive / Dynamic Prog. |
| OSKI | Sparse Linear Algebra | Exhaustive + Heuristic |
| Periscope | HPC | Exhaustive + Nelder-Mead |
| Active Harmony | Runtime System | Nelder-Mead |
| PATUS | Stencil Computations | Nelder-Mead or Evolutionary |
| Sepya | Stencil Computations | Random-Restart Gradient Ascent |
| Dynamic Knobs | Runtime System | Control Theory |
| Milepost GCC / cTuning | Compiler | IID Model + Central DB |
| SEEC / Heartbeats | Runtime System | Control Theory |
| Insieme | Compiler | Differential Evolution |
| PetaBricks | Programming Language | Bottom-up Evolutionary |
| SPIRAL | DSP Algorithms | Pareto Active Learning |

- Simple techniques (exhaustive, hill climbers, etc) are popular
  - No single technique is best for all problems
- Representations are often just integers/floats/booleans

# Limits of Current Approaches

- We believe simple techniques limit the scope and efficiency of autotuning
- A hill climber works great for a block size, but fails for more complex applications
- Many users of autotuning work hard to prune their search spaces to fit techniques such as exhaustive search

# Limits of Current Approaches

- We believe simple techniques limit the scope and efficiency of autotuning
- A hill climber works great for a block size, but fails for more complex applications
- Many users of autotuning work hard to prune their search spaces to fit techniques such as exhaustive search

- Real problems have large search spaces

# Over $10^{806}$ Combinations of GCC Optimizations

```
g++ apps/raytracer.cpp -o ./raytracer_c -O3 -fno-align-functions -fno-align-loops -fasynchronous-unwind-tables -fbranch-count-reg -fbranch-probabilities
-fno-branch-target-load-optimize -fbtr-bb-exclusive -fno-combine-stack-adjustments -fno-common -fcompare-elim -fcrossjumping -fcse-follow-jumps
-fcx-fortran-rules -fcx-limited-range -fdata-sections -fno-dce -fdelete-null-pointer-checks -fno-devirtualize -fno-dse -fearly-inlining -fexceptions
-ffinite-math-only -fforward-propagate -fgcse -fgcse-after-reload -fno-gcse-las -fno-graphite-identity -fno-if-conversion2 -fno-inline-functions
-fno-inline-small-functions -fno-ipa-cp -fno-ipa-matrix-reorg -fno-ipa-profile -fno-ipa-pta -fipa-pure-const -fipa-reference -fno-ipa-sra -fno-ivopts
-fno-loop-block -fno-loop-flatten -floop-interchange -fno-loop-parallelize-all -floop-strip-mine -fmath-errno -fno-merge-all-constants -fno-modulo-sched
-fno-non-call-exceptions -fno-optimize-sibling-calls -fno-optimize-strlen -fpeel-loops -fpeephole -fno-peephole2 -fno-predictive-commoning
-fno-prefetch-loop-arrays -fno-reg-struct-return -fno-regmove -frename-registers -fno-reorder-blocks -freorder-blocks-and-partition -freorder-functions
-fno-rerun-cse-after-loop -fno-rounding-math -fno-rtti -fno-sched-critical-path-heuristic -fno-sched-dep-count-heuristic -fno-sched-group-heuristic
-fno-sched-interblock -fno-sched-pressure -fsched-rank-heuristic -fsched-spec-insn-heuristic -fsched-spec-load -fno-sched-stalled-insns -fsched-stalled-insns-dep
-fno-sched2-use-superblocks -fno-schedule-insns -fschedule-insns2 -fno-sel-sched-pipelining -fno-sel-sched-pipelining-outer-loops -fsel-sched-reschedule-pipelined
-fno-short-wchar -fno-shrink-wrap -fsignaling-nans -fsingle-precision-constant -fno-split-ivs-in-unroller -fstrict-enums -fno-thread-jumps
-ftrapping-math -fno-trapv -fno-tree-builtin-call-dce -fno-tree-ccp -fno-tree-copy-prop -ftree-copyrename -fno-tree-cselim -fno-tree-dce -ftree-dse
-fno-tree-forwprop -ftree-fre -ftree-loop-distribute-patterns -fno-tree-loop-distribution -ftree-loop-if-convert -fno-tree-loop-if-convert-stores
-fno-tree-loop-ivcanon -ftree-pta -fno-tree-reassoc -fno-tree-scev-cprop -fno-tree-slp-vectorize -ftree-sra -ftree-switch-conversion -fno-tree-ter
-fno-tree-vectorize -ftree-vrp -fno-unit-at-a-time -fno-unroll-all-loops -fno-unroll-loops -funsafe-loop-optimizations -funwind-tables -fno-var-tracking
-fvar-tracking-assignments-toggle -fno-var-tracking-uninit -fno-vect-cost-model -fno-vpt -fweb -fwhole-program -fwrapv --param=align-loop-iterations=16
--param=align-threshold=28 --param=allow-load-data-races=1 --param=allow-packed-load-data-races=1 --param=allow-packed-store-data-races=0
--param=allow-store-data-races=1 --param=case-values-threshold=3 --param=comdat-sharing-probability=14 --param=cxx-max-namespaces-for-diagnostic-help=1008
--param=early-inlining-insns=19 --param=gcse-after-reload-critical-fraction=15 --param=gcse-after-reload-partial-fraction=10 --param=gcse-cost-distance-ratio=14
--param=gcse-unrestricted-cost=5 --param=ggc-min-expand=66 --param=ggc-min-heapsize=15449 --param=graphite-max-bbs-per-function=248 --param=graphite-max-nb-scop-params=10
--param=hot-bb-count-ws-permille=271 --param=hot-bb-frequency-fraction=2357 --param=inline-min-speedup=36 --param=inline-unit-growth=26 --param=integer-share-limit=511
--param=ipa-cp-eval-threshold=222 --param=ipa-cp-loop-hint-bonus=18 --param=ipa-cp-value-list-size=18 --param=ipa-max-agg-items=13 --param=ipa-sra-ptr-growth-factor=6
--param=ipcp-unit-growth=3 --param=ira-loop-reserved-regs=8 --param=ira-max-conflict-table-size=261 --param=ira-max-loops-num=25 --param=iv-always-prune-cand-set-bound=17
--param=iv-consider-all-candidates-bound=26 --param=iv-max-considered-uses=85 --param=l1-cache-line-size=128 --param=l1-cache-size=24 --param=l2-cache-size=356
--param=large-function-growth=237 --param=large-function-insns=4444 --param=large-stack-frame=431 --param=large-stack-frame-growth=250 --param=large-unit-insns=2520
--param=lim-expensive=10 --param=loop-block-tile-size=40 --param=loop-invariant-max-bbs-in-loop=2500 --param=loop-max-datarefs-for-datadeps=816
--param=lto-min-partition=261 --param=lto-partitions=96 --param=max-average-unrolled-insns=22 --param=max-completely-peel-loop-nest-depth=18
--param=max-completely-peel-times=31 --param=max-completely-peeled-insns=325 --param=max-crossjump-edges=30 --param=max-cse-insns=251 --param=max-cse-path-length=8
--param=max-cselib-memory-locations=1202 --param=max-delay-slot-insn-search=137 --param=max-delay-slot-live-search=84 --param=max-dse-active-local-stores=1250
--param=max-early-inliner-iterations=2 --param=max-fields-for-field-sensitive=0 --param=max-gcse-insertion-ratio=50 --param=max-gcse-memory=13107200
--param=max-goto-duplication-insns=15 --param=max-grow-copy-bb-insns=23 --param=max-hoist-depth=101 --param=max-inline-insns-auto=43 --param=max-inline-insns-recursive=126
--param=max-inline-insns-recursive-auto=135 --param=max-inline-insns-single=421 --param=max-inline-recursive-depth=24 --param=max-inline-recursive-depth-auto=28
--param=max-iterations-computation-cost=24 --param=max-iterations-to-track=253 --param=max-jump-thread-duplication-stmts=21 --param=max-last-value-rtl=2794
--param=max-modulo-backtrack-attempts=14 --param=max-once-peeled-insns=105 --param=max-partial-antic-length=25 --param=max-peel-branches=84
--param=max-peel-times=23 --param=max-peeled-insns=25 --param=max-pending-list-length=10 --param=max-pipeline-region-blocks=44 --param=max-pipeline-region-insns=578
--param=max-predicted-iterations=28 --param=max-reload-search-insns=356 --param=max-sched-extend-regions-iters=1 --param=max-sched-insn-conflict-delay=1
--param=max-sched-ready-insns=101 --param=max-sched-region-blocks=15 --param=max-sched-region-insns=36 --param=max-slsr-cand-scan=12 --param=max-stores-to-sink=2
--param=max-tail-merge-comparisons=24 --param=max-tail-merge-iterations=1 --param=max-tracked-strlens=351 --param=max-unroll-times=26 --param=max-unrolled-insns=570
--param=max-unswitch-insns=17 --param=max-unswitch-level=11 --param=max-variable-expansions-in-unroller=0 --param=max-vartrack-expr-depth=14
--param=max-vartrack-reverse-op-size=15 --param=max-vartrack-size=12500164 --param=min-crossjump-insns=18 --param=min-inline-recursive-probability=9
--param=min-insn-to-prefetch-ratio=23 --param=min-spec-prob=15 --param=min-vect-loop-bound=2 --param=omega-eliminate-redundant-constraints=0
--param=omega-hash-table-size=138 --param=omega-max-eqs=43 --param=omega-max-geqs=68 --param=omega-max-keys=378 --param=omega-max-vars=32
--param=omega-max-wild-cards=55 --param=partial-inlining-entry-probability=68 --param=predictable-branch-outcome=0 --param=prefetch-latency=115
--param=prefetch-min-insn-to-mem-ratio=2 --param=sccvn-max-alias-queries-per-access=2543 --param=sccvn-max-scc-size=2504 --param=scev-max-expr-complexity=32
--param=scev-max-expr-size=45 --param=sched-mem-true-dep-cost=0 --param=sched-pressure-algorithm=1 --param=sched-spec-prob-cutoff=79 --param=sched-state-edge-prob-cutoff=2
--param=selsched-insns-to-rename=6 --param=selsched-max-lookahead=14 --param=selsched-max-sched-times=1 --param=simultaneous-prefetches=9
--param=sink-frequency-threshold=53 --param=slp-max-insns-in-bb=279 --param=sms-dfa-history=3 --param=sms-loop-average-count-threshold=2
--param=sms-max-ii-factor=35 --param=sms-min-sc=3 --param=ssp-buffer-size=13 --param=switch-conversion-max-branch-ratio=2 --param=tm-max-aggregate-size=32
--param=tracer-dynamic-coverage=66 --param=tracer-dynamic-coverage-feedback=46 --param=tracer-max-code-growth=200 --param=tracer-min-branch-probability=82
--param=tracer-min-branch-probability-feedback=70 --param=tracer-min-branch-ratio=21 --param=tree-reassoc-width=2 --param=uninit-control-dep-attempts=415
--param=use-canonical-types=0 --param=vect-max-version-for-alias-checks=11 --param=vect-max-version-for-alignment-checks=23
```

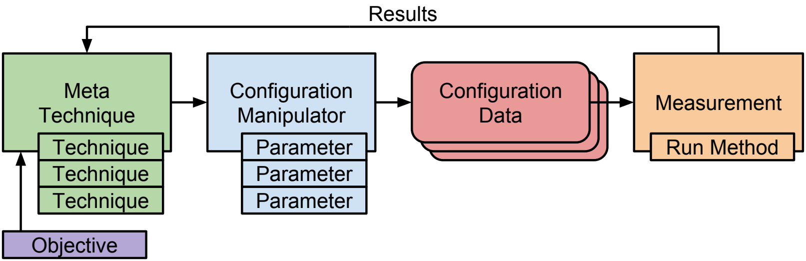
# Large Search Spaces are a Challenge

| Project | Benchmark | Possible Configurations |
|---|---|---|
| GCC/G++ Flags | *all* | $10^{806}$ |
| Halide | Blur | $10^{25}$ |
| Halide | Wavelet | $10^{32}$ |
| Halide | Bilateral | $10^{176}$ |
| HPL | *n/a* | $10^{9.9}$ |
| PetaBricks | Poisson | $10^{3657}$ |
| PetaBricks | Sort | $10^{90}$ |
| PetaBricks | Strassen | $10^{188}$ |
| PetaBricks | TriSolve | $10^{1559}$ |
| Stencil | *all* | $10^{6.5}$ |
| Unitary | *n/a* | $10^{21}$ |
| Mario | *n/a* | $10^{6328}$ |

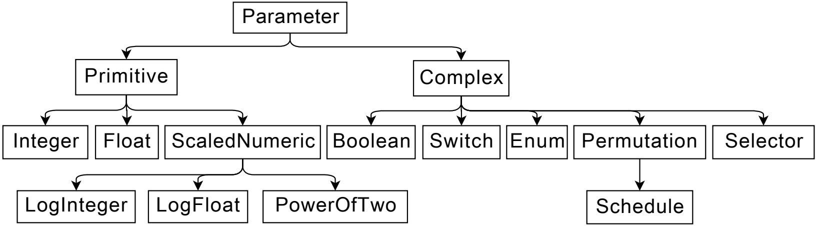
# OpenTuner's General Representation

- Large search spaces do not mean haphazard ones
- Choosing the right representation is critical
- OpenTuner allows programmers to easily express structured search spaces
  - Supports complex parameter types such as permutations, schedules, mappings
  - User defined parameter types

# OpenTuner Model



# OpenTuner Configuration Manipulator Parameters

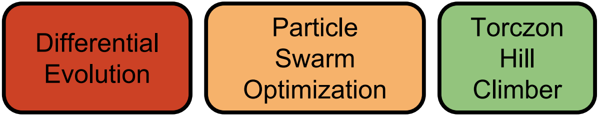



- Hierarchical structure of parameters, user defined parameter types can be added at any point
- Primitive parameters behave like bounded integers or floats
- Complex parameters have a set of stochastic mutation operators
- Technique-specific operators

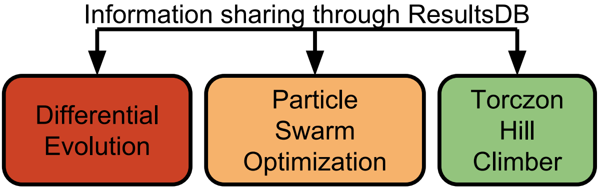
# Ensembles of Techniques

- OpenTuner contains many techniques such as:
  - Differential Evolution
  - Genetic Algorithms
  - Greedy Mutation
  - Multi-armed Bandit
  - Nelder Mead
  - Partial Swarm Optimization
  - Pattern Search
  - Pseudo Annealing
  - Torczon
- Uses ensembles of techniques to provide robustness to different search spaces

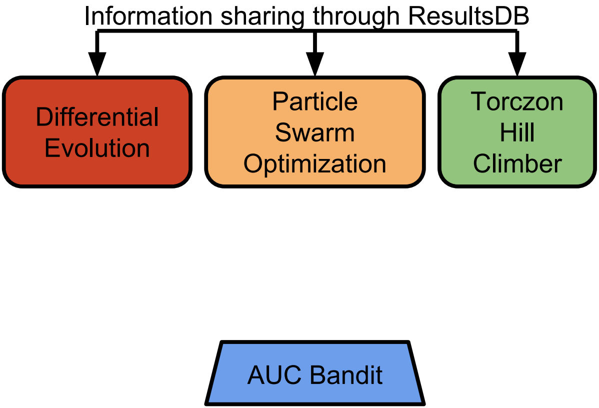
# Ensembles of Techniques in OpenTuner



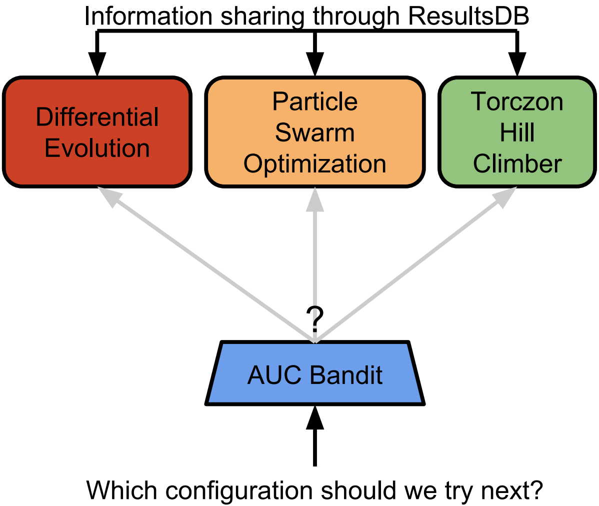
# Ensembles of Techniques in OpenTuner



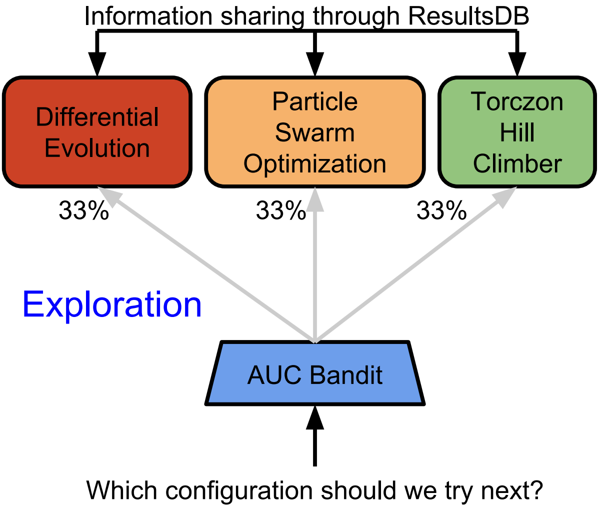
# Ensembles of Techniques in OpenTuner



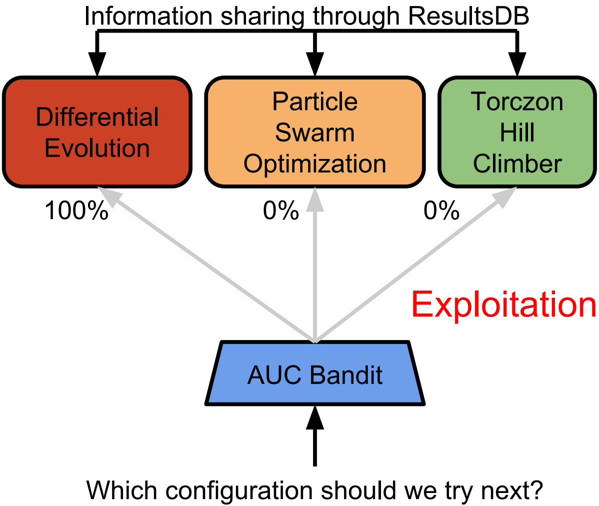
# Ensembles of Techniques in OpenTuner



# Ensembles of Techniques in OpenTuner



# Ensembles of Techniques in OpenTuner



# AUC Bandit[1]

$$\arg\max_t (AUC_t + C\sqrt{\frac{2 \lg |H|}{H_t}})$$

- $|H|$ is the length of the sliding history window
- $H_t$ is the number of times the technique has been used in that history window,
- $C$ is a constant controlling the exploration/exploitation trade-off
- $AUC_t$ is the credit assignment term

[1]Based on strategy in Fialho PPSN'10

# AUC Bandit[1]

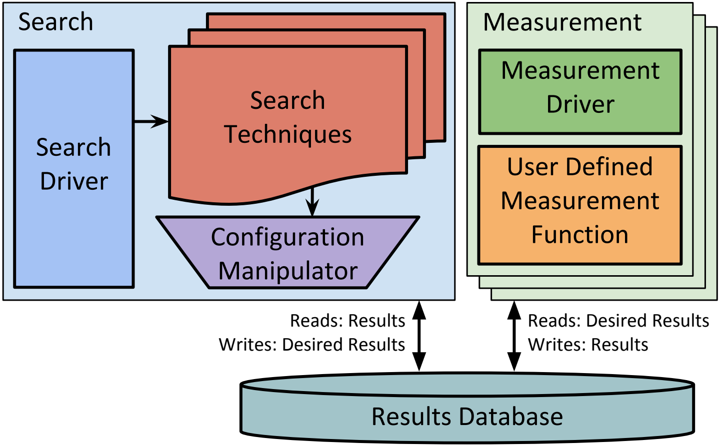
Exploitation Exploration

$$\arg\max_t \left( AUC_t + C\sqrt{\frac{2\lg|H|}{H_t}} \right)$$

- $|H|$ is the length of the sliding history window
- $H_t$ is the number of times the technique has been used in that history window,
- $C$ is a constant controlling the exploration/exploitation trade-off
- $AUC_t$ is the credit assignment term

[1] Based on strategy in Fialho PPSN'10

# OpenTuner System Overview



# Conclusions

- A lot of performance is left on the floor due to poorly optimized programs
- OpenTuner makes state of the art machine learning accessible to all
  - Extensible configuration representation
  - Ensembles of techniques
- Conventional wisdom underestimates the size tractable search spaces
- However, choosing the right representation is critical to successful autouners

http://opentuner.org/

# Today's Agenda

- 08:30 Welcome and broader context (Saman Amarasinghe)
- 08:40 Introduction to OpenTuner (Jason Ansel)
- **09:10 Search techniques (Kalyan Veeramachaneni)**
- 09:35 In depth example (Jeffrey Bosboom)
- 10:00 Break
- 10:15 Applications
  - Halide (Jonathan Ragan-Kelley)
  - SEJITS (Chick Markley)
  - JVM optimization (Tharindu Rusira)
- 11:00 Hands on session (Shoaib Kamil)
- 11:45 Discussion

# Backup Slides: Mario

# OpenTuner Can Play Super Mario Bros!

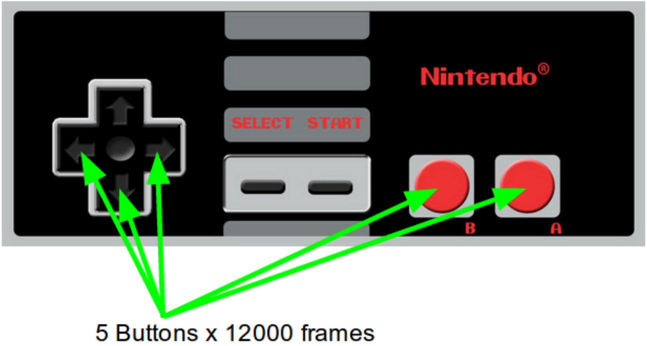

(Video [2])

[2] `http://youtu.be/pTi_tHpj6Ow`

# OpenTuner Can Play Super Mario Bros!

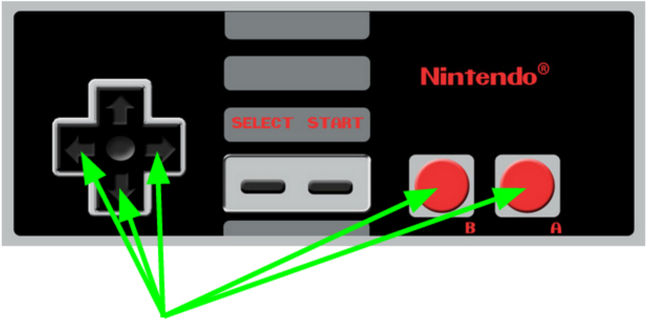
- Only feedback is number of pixels moved to the right
  - e.g. approximately 1500 pixels for first pit
- OpenTuner doesn't see the screen
- Super Mario Bros is deterministic, single run suffices

# Naive Representation




[3] http://youtu.be/nyYdq1jJQrw

# Naive Representation




5 Buttons x 12000 frames

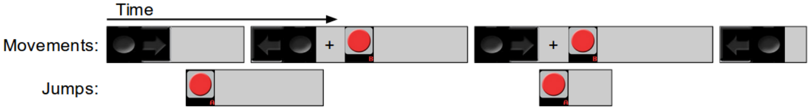
- Bad, because most configurations make no sense.
- Just mashing random buttons.
- Doesn't work at all (Video [3]).

[3] http://youtu.be/nyYdq1jJQrw

# Better Representation

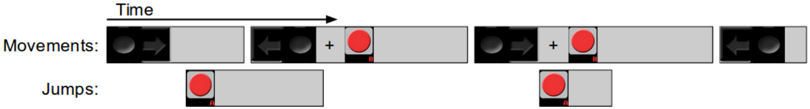

- Movements (list):
  - Direction (left, right, run left, or run right)
  - Duration (frames)

# Better Representation




- Movements (list):
  - Direction (left, right, run left, or run right)
  - Duration (frames)
- Jumps (list):
  - Start frame
  - Duration (frames)

# Better Representation

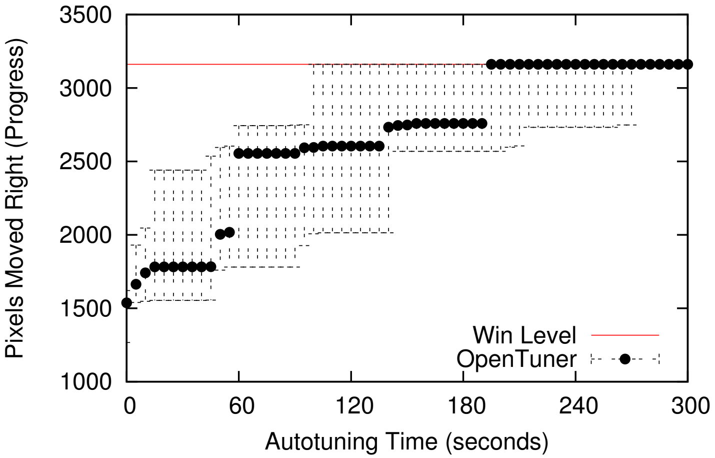



- Movements (list):
  - Direction (left, right, run left, or run right)
  - Duration (frames)
- Jumps (list):
  - Start frame
  - Duration (frames)

Choosing the right representation is critical

- Search space size $10^{6328}$
- Winning run found in 13641 ($\approx 10^4$) attempts
- Under 5 minutes of training time

# Super Mario Bros Results



# A Final Video [4]

- OpenTuner learning to play Super Mario Bros
- Every run that achieves a high score
- Runs that don't make improvements are skipped
- Run # in top left caption

- Thanks!

`http://opentuner.org/`
`pip install opentuner`

Try OpenTuner today!

[4] `http://youtu.be/O5IK9f2nBsE`

# Backup Slides: Halide

# OpenTuner Generating Halide Schedules

- A domain specific language for image processing and photography
- Used for camera pipeline in Google Glass, HDR+ in Android, some filters in Photoshop
- Separate algorithm language and scheduling language
- We use OpenTuner to generate the scheduling language

# Simple Halide Example

Algorithm:

```
ImageParam input(UInt(16), 2);
Func a("a"), a("b"), a("c");
Var x("x"), y("y");

a(x, y) = input(x, y);
b(x, y) = a(x, y);
c(x, y) = b(x, y);
```

# Simple Halide Example

Algorithm:

```
ImageParam input(UInt(16), 2);
Func a("a"), a("b"), a("c");
Var x("x"), y("y");

a(x, y) = input(x, y);
b(x, y) = a(x, y);
c(x, y) = b(x, y);
```

OpenTuner Generated Schedule:

```
Var x0, y1, x2, x4, y5;
a.split(x, x, x0, 4)
 .split(y, y, y1, 16)
 .reorder(y1, x0, y, x)
 .vectorize(y1, 4)
 .compute_at(b, y);
b.split(x, x, x2, 64)
 .reorder(x2, x, y)
 .reorder_storage(y, x)
 .vectorize(x2, 8)
 .compute_at(c, x4);
c.split(x, x, x4, 8)
 .split(y, y, y5, 2)
 .reorder(x4, y5, y, x)
 .parallel(x)
 .compute_root();
```

# Simple Halide Example

Algorithm:

```
ImageParam input(UInt(16), 2);
Func a("a"), a("b"), a("c");
Var x("x"), y("y");

a(x, y) = input(x, y);
b(x, y) = a(x, y);
c(x, y) = b(x, y);
```

Complex schedules:

- Split
- Reorder / reorder storage
- Vectorize / Parallel
- Compute_at / compute_root

OpenTuner Generated Schedule:

```
Var x0, y1, x2, x4, y5;
a.split(x, x, x0, 4)
 .split(y, y, y1, 16)
 .reorder(y1, x0, y, x)
 .vectorize(y1, 4)
 .compute_at(b, y);
b.split(x, x, x2, 64)
 .reorder(x2, x, y)
 .reorder_storage(y, x)
 .vectorize(x2, 8)
 .compute_at(c, x4);
c.split(x, x, x4, 8)
 .split(y, y, y5, 2)
 .reorder(x4, y5, y, x)
 .parallel(x)
 .compute_root();
```

# Simple Halide Example

Algorithm:

```
ImageParam input(UInt(16), 2);
Func a("a"), a("b"), a("c");
Var x("x"), y("y");

a(x, y) = input(x, y);
b(x, y) = a(x, y);
c(x, y) = b(x, y);
```

Complex schedules:

- Split
- Reorder / reorder storage
- Vectorize / Parallel
- Compute_at / compute_root

OpenTuner Generated Schedule:

```
Var x0, y1, x2, x4, y5;
a.split(x, x, x0, 4)
 .split(y, y, y1, 16)
 .reorder(y1, x0, y, x)
 .vectorize(y1, 4)
 .compute_at(b, y);
b.split(x, x, x2, 64)
 .reorder(x2, x, y)
 .reorder_storage(y, x)
 .vectorize(x2, 8)
 .compute_at(c, x4);
c.split(x, x, x4, 8)
 .split(y, y, y5, 2)
 .reorder(x4, y5, y, x)
 .parallel(x)
 .compute_root();
```

# Simplified Schedules (Placement Only)

Schedule:

```
a.compute_at(b, y)
b.compute_at(c, x)
c.compute_root()
```

# Simplified Schedules (Placement Only)

Schedule:

```
a.compute_at(b, y)
b.compute_at(c, x)
c.compute_root()
```

Logical Loop Structure:

# Simplified Schedules (Placement Only)

Logical Loop Structure:

```
for c_x:




  for c_y:
    compute_c()
```

Schedule:

```
a.compute_at(b, y)
b.compute_at(c, x)
c.compute_root()
```

# Simplified Schedules (Placement Only)

Schedule:

```
a.compute_at(b, y)
b.compute_at(c, x)
c.compute_root()
```

Logical Loop Structure:

```
for c_x:
  for b_x:
    for b_y:



      compute_b()
  for c_y:
    compute_c()
```

# Simplified Schedules (Placement Only)

Logical Loop Structure:

```
for c_x:
  for b_x:
    for b_y:
      for a_x:
        for a_y:
          compute_a()
      compute_b()
  for c_y:
    compute_c()
```

Schedule:

```
a.compute_at(b, y)
b.compute_at(c, x)
c.compute_root()
```

# Simplified Schedules (Placement Only)

Schedule:

```
a.compute_at(b, y)
b.compute_at(c, x)
c.compute_root()
```

Logical Loop Structure:

```
for c_x:
  for b_x:
    for b_y:
      for a_x:
        for a_y:
          compute_a()
      compute_b()
  for c_y:
    compute_c()
```

Resulting Code:

```
for x:
  for y:
    tmp_a = input[x, y]
    tmp_b[y] = tmp_a
  for y:
    output[x, y] = tmp_b[y]
```

# Naive Halide Representation

Based on Halide scheduling language:

```
a.compute_at(b, y)
b.compute_at(c, x)
c.compute_root()
```

# Naive Halide Representation

Based on Halide scheduling language:

```
a.compute_at(b, y)
b.compute_at(c, x)
c.compute_root()
```

- 8 possible placements:
  - `compute_at(a, x)`,
  - `compute_at(a, y)`,
  - `compute_at(b, x)`,
  - `compute_at(b, y)`,
  - `compute_at(c, x)`,
  - `compute_at(c, y)`,
  - `compute_root()`,
  - *inline*

# Naive Halide Representation

Based on Halide scheduling language:

```
a.compute_at(b, y)
b.compute_at(c, x)
c.compute_root()
```

- 8 possible placements:
  - compute_at(a, x),
  - compute_at(a, y),
  - compute_at(b, x),
  - compute_at(b, y),
  - compute_at(c, x),
  - compute_at(c, y),
  - compute_root(),
  - *inline*
- 3 computations that must be placed (a, b, c):
- 512 possible schedules

# Naive Representation Does Not Work

- Naive representation works for simple halide programs
- Fails completely for more complex programs

# Naive Representation Does Not Work

- Naive representation works for simple halide programs
- Fails completely for more complex programs
- 474 of 512 schedules are invalid
  - Callgraph orderings not respected
  - Exponentially worse with larger programs
- Poor locality
  - Small changes move large subtrees around

```
for c_x:
  for b_x:
    for b_y:
      for a_x:
        for a_y:
          compute_a()
      compute_b()
  for c_y:
    compute_c()
```

# Better Representation

```
for c_x:
   for b_x:
      for b_y:
         for a_x:
            for a_y:
               compute_a()
         compute_b()
   for c_y:
      compute_c()
```

c_x
b_x
x_y
a_x
a_y
compute_a
compute_b
c_y
compute_c

- Representation based logical loop structure
- Loop structure can be reconstructed from token order
- Representation is a *permutation* of tokens that:

# Better Representation

```
for c_x:
  for b_x:
    for b_y:
      for a_x:
        for a_y:
          compute_a()
      compute_b()
  for c_y:
    compute_c()
```

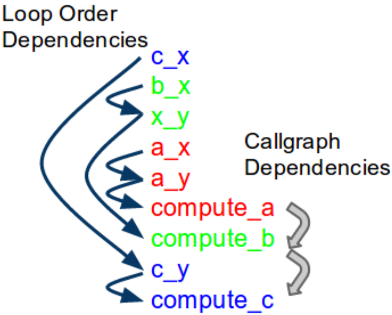
c_x
b_x
x_y
a_x
a_y
compute_a
compute_b
c_y
compute_c

Callgraph
Dependencies

- Representation based logical loop structure
- Loop structure can be reconstructed from token order
- Representation is a *permutation* of tokens that:
  - Respects callgraph orderings

# Better Representation

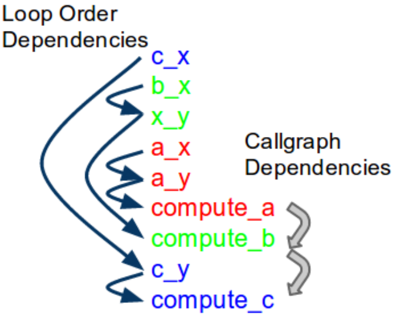
```
for c_x:
  for b_x:
    for b_y:
      for a_x:
        for a_y:
          compute_a()
      compute_b()
  for c_y:
    compute_c()
```




- Representation based logical loop structure
- Loop structure can be reconstructed from token order
- Representation is a *permutation* of tokens that:
  - Respects callgraph orderings
  - Respects loop orderings

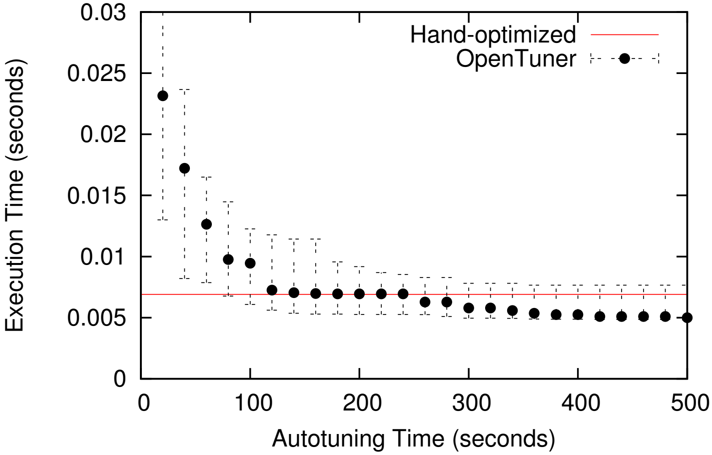
# Better Representation

```
for c_x:
  for b_x:
    for b_y:
      for a_x:
        for a_y:
          compute_a()
      compute_b()
  for c_y:
    compute_c()
```




- Representation based logical loop structure
- Loop structure can be reconstructed from token order
- Representation is a *permutation* of tokens that:
  - Respects callgraph orderings
  - Respects loop orderings
- Handling of some subtle corner cases and `reorder()` discussed in the paper

# Halide Blur



Hand-optimized
OpenTuner
Execution Time (seconds)
Autotuning Time (seconds)
0
0.005
0.01
0.015
0.02
0.025
0.03
0
100
200
300
400
500

# Halide Bilateral Grid