OPENACS AND TCL/TK CONFERENCE 2026, JULY 17-18

- Andreas Kupries
- AKTIVE
  - AK Tcl Image Vector Extension
  - MIT/BSD Licensed
  - Inspired by VIPS
  - Mixes C and Tcl via Critcl

AK TCL IMAGE VECTOR EXTENSION

- Last year: Introduced by use, light on internals

AK TCL IMAGE VECTOR EXTENSION

- Last year: Introduced by use, light on internals
  - See [PDF] [ZIP] [WU Lecturecast] (Entire conference)

https://core.tcl-lang.org/akupries/aktive/doc/trunk/doc/presentations/eurotcl-2025-bologna.pdf

https://core.tcl-lang.org/akupries/aktive/doc/trunk/doc/presentations/eurotcl-2025-bologna.zip

https://learn.wu.ac.at/eurotcl2025/lecturecasts/753988258?m=delivery

https://learn.wu.ac.at/eurotcl2025/lecturecasts/?page=1

AK TCL IMAGE VECTOR EXTENSION

- Last year: Introduction by use, light on internals
- **Today**: A deeper look at the internals

AK TCL IMAGE VECTOR EXTENSION

- Last year: Introduction by use, light on internals
- **Today**: A deeper look at the internals
  - Demand-driven: Lazy construction

AK TCL IMAGE VECTOR EXTENSION

- Last year: Introduction by use,
  light on internals
- **Today**: A deeper look at the internals
  - Demand-driven: Lazy construction
    - Data structures (definition, run)

AK TCL IMAGE VECTOR EXTENSION

- Last year: Introduction by use,
  light on internals
- **Today**: A deeper look at the internals
  - Demand-driven: Lazy construction
    - Data structures (definition, run)
  - Horizontally Threaded Execution

AK TCL IMAGE VECTOR EXTENSION

- Last year: Introduction by use,
  light on internals
- **Today**: A deeper look at the internals
  - Demand-driven: Lazy construction
    - Data structures (definition, run)
  - Horizontally Threaded Execution
    - Thread coordination, ordered and not

AK TCL IMAGE VECTOR EXTENSION

- Last year: Introduction by use,
  light on internals
- **Today**: A deeper look at the internals
  - Demand-driven: Lazy construction
    - Data structures (definition, run)
  - Horizontally Threaded Execution
    - Thread coordination, ordered and not
- Benchmarking & Performance

AK TCL IMAGE VECTOR EXTENSION

- Last year: Introduction by use, light on internals
- **Today**: A deeper look at the internals
  - Demand-driven: Lazy construction
    - Data structures (definition, run)
  - Horizontally Threaded Execution
    - Thread coordination, ordered and not
- Benchmarking & Performance
  - Loop unrolling

AK TCL IMAGE VECTOR EXTENSION

- Last year: Introduction by use,
  light on internals
- **Today**: A deeper look at the internals
  - Demand-driven: Lazy construction
    - Data structures (definition, run)
  - Horizontally Threaded Execution
    - Thread coordination, ordered and not
- Benchmarking & Performance
  - Loop unrolling
  - (SIMD) Vectorization

AK TCL IMAGE VECTOR EXTENSION

LAZY CONSTRUCTION

- Image creation builds **definition** structures

AK TCL IMAGE VECTOR EXTENSION

LAZY CONSTRUCTION

- Image creation builds **definition** structures
  - Linear pipe

AK TCL IMAGE VECTOR EXTENSION

LAZY CONSTRUCTION

- Image creation builds **definition** structures
  - Linear pipe
  - Tree

AK TCL IMAGE VECTOR EXTENSION

LAZY CONSTRUCTION

- Image creation builds **definition** structures
  - Linear pipe
  - Tree
  - Directed acyclic graph

AK TCL IMAGE VECTOR EXTENSION

LAZY CONSTRUCTION

- Image materialization builds **execution** structures

AK TCL IMAGE VECTOR EXTENSION

LAZY CONSTRUCTION

- Image materialization builds **execution** structures
  - ... computes things

AK TCL IMAGE VECTOR EXTENSION

LAZY CONSTRUCTION

- Image materialization builds **execution** structures
  - ... computes things
  - ... and tears the execution structures down again

AK TCL IMAGE VECTOR EXTENSION

LAZY CONSTRUCTION

- Image materialization builds **execution** structures
  - ... computes things
  - ... and tears the execution structures down again
  - **The definition structures stay**

AK TCL IMAGE VECTOR EXTENSION

LAZY CONSTRUCTION

- Examples of materializators

AK TCL IMAGE VECTOR EXTENSION

LAZY CONSTRUCTION

- Examples of materializators
  - Saving pixels to file, channel, memory, ...

AK TCL IMAGE VECTOR EXTENSION

LAZY CONSTRUCTION

- Examples of materializators
  - Saving pixels to file, channel, memory, ...
  - Computing whole-image statistics

AK TCL IMAGE VECTOR EXTENSION

PREFACE: INSPECTION


```
aktive format as d2         $image
aktive format as markdown   $image
aktive format as tclscript $image
```

AK TCL IMAGE VECTOR EXTENSION

PREFACE: INSPECTION


```
aktive format as d2         $image
aktive format as markdown   $image
aktive format as tclscript $image
```

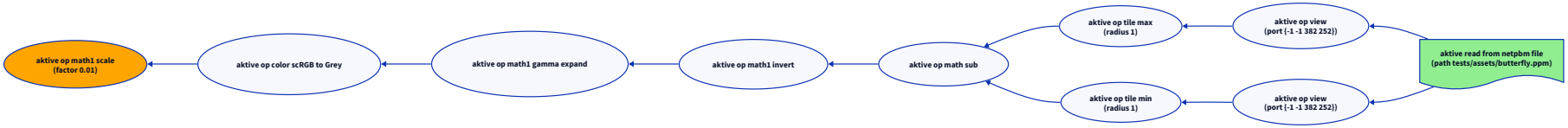
```
set file1 [aktive read from netpbm file path ...];# FO(2): tmp2,tmp
set tmp2 [aktive op view $file1 port {-1 -1 382 252}]
set tmp3 [aktive op tile max $tmp2 radius 1]
set tmp4 [aktive op view $file1 port {-1 -1 382 252}]
set tmp5 [aktive op tile min $tmp4 radius 1]
set tmp6 [aktive op math sub $tmp3 $tmp5]
set tmp7 [aktive op math1 invert $tmp6]
set tmp8 [aktive op math1 gamma expand $tmp7]
set tmp9 [aktive op color scRGB to Grey $tmp8]
set result [aktive op math1 scale $tmp9 factor 0.01]
```

AK TCL IMAGE VECTOR EXTENSION

PREFACE: INSPECTION


```
aktive format as d2          $image
aktive format as markdown    $image
aktive format as tclscript $image
```

Charcoal effect

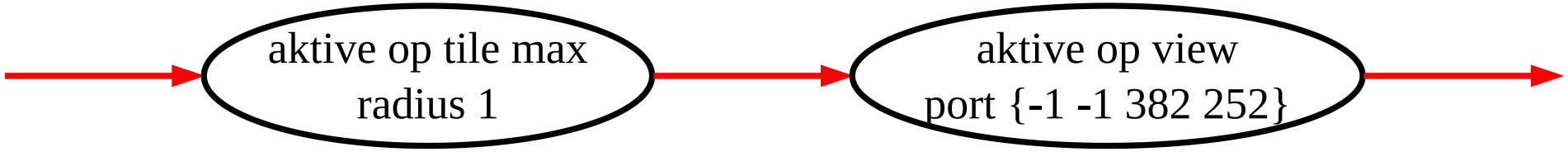


full size (assets/charcoal.svg)

AK TCL IMAGE VECTOR EXTENSION

DEFINITION STRUCTURES

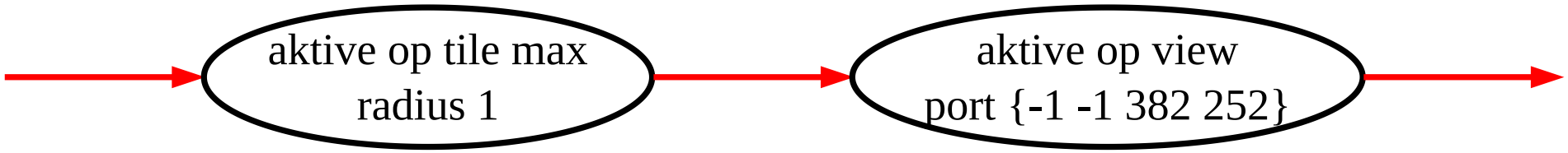
Shoutout: Pikchr @ https://pikchr.org



AK TCL IMAGE VECTOR EXTENSION

DEFINITION STRUCTURES

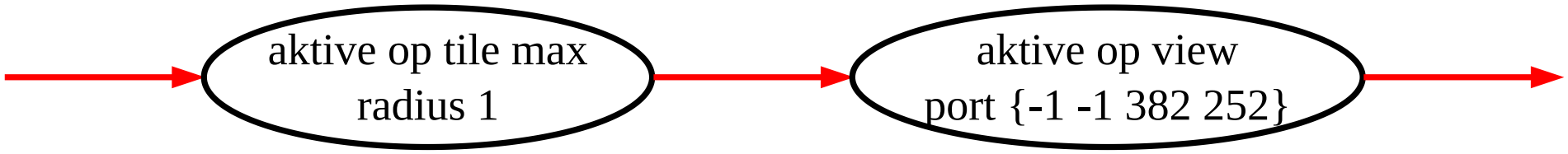
Shoutout: Pikchr @ https://pikchr.org



- type (function vectors, parameter defs)
- parameters
- srcs
- domain (geometry)
- state

AK TCL IMAGE VECTOR EXTENSION

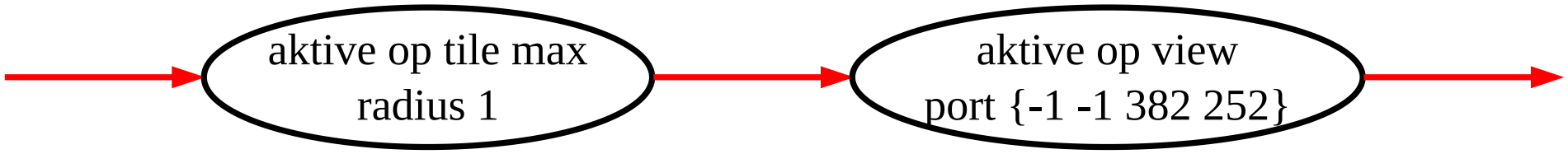
DEFINITION STRUCTURES



- type (function vectors, parameter defs)
- content (parameters, srcs, domain, state)

AK TCL IMAGE VECTOR EXTENSION

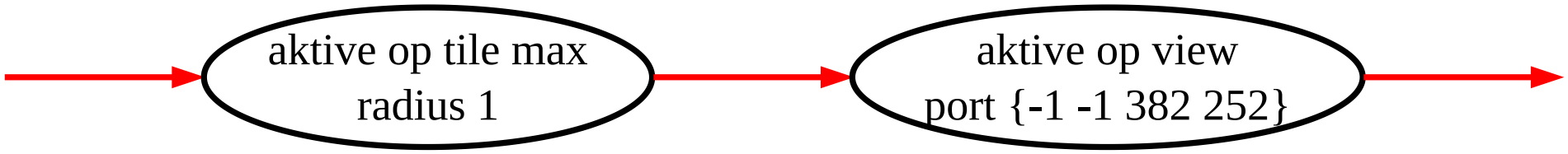
DEFINITION STRUCTURES



- type (function vectors, parameter defs)
- content (parameters, srcs, domain, state)
- metadata

AK TCL IMAGE VECTOR EXTENSION

DEFINITION STRUCTURES

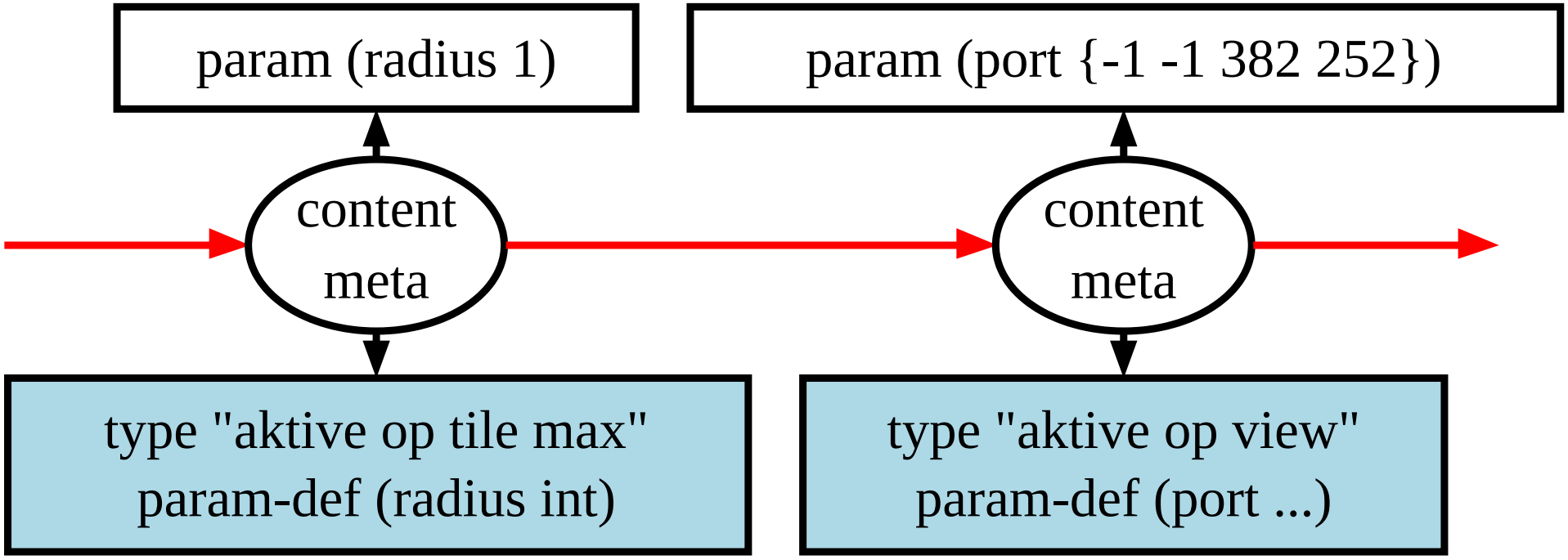


- type (function vectors, parameter defs)
- content (parameters, srcs, domain, state)
- metadata
- refcount

AK TCL IMAGE VECTOR EXTENSION

DEFINITION STRUCTURES

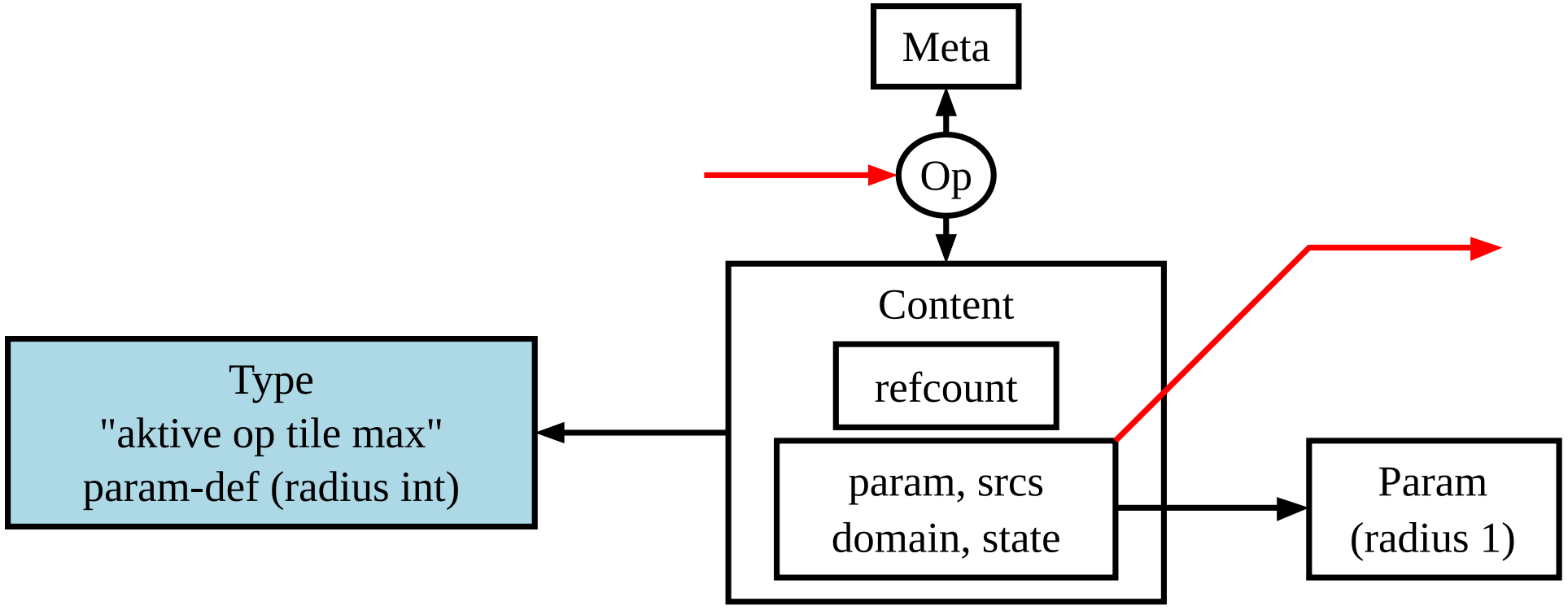
Types and Parameters



AK TCL IMAGE VECTOR EXTENSION

DEFINITION STRUCTURES

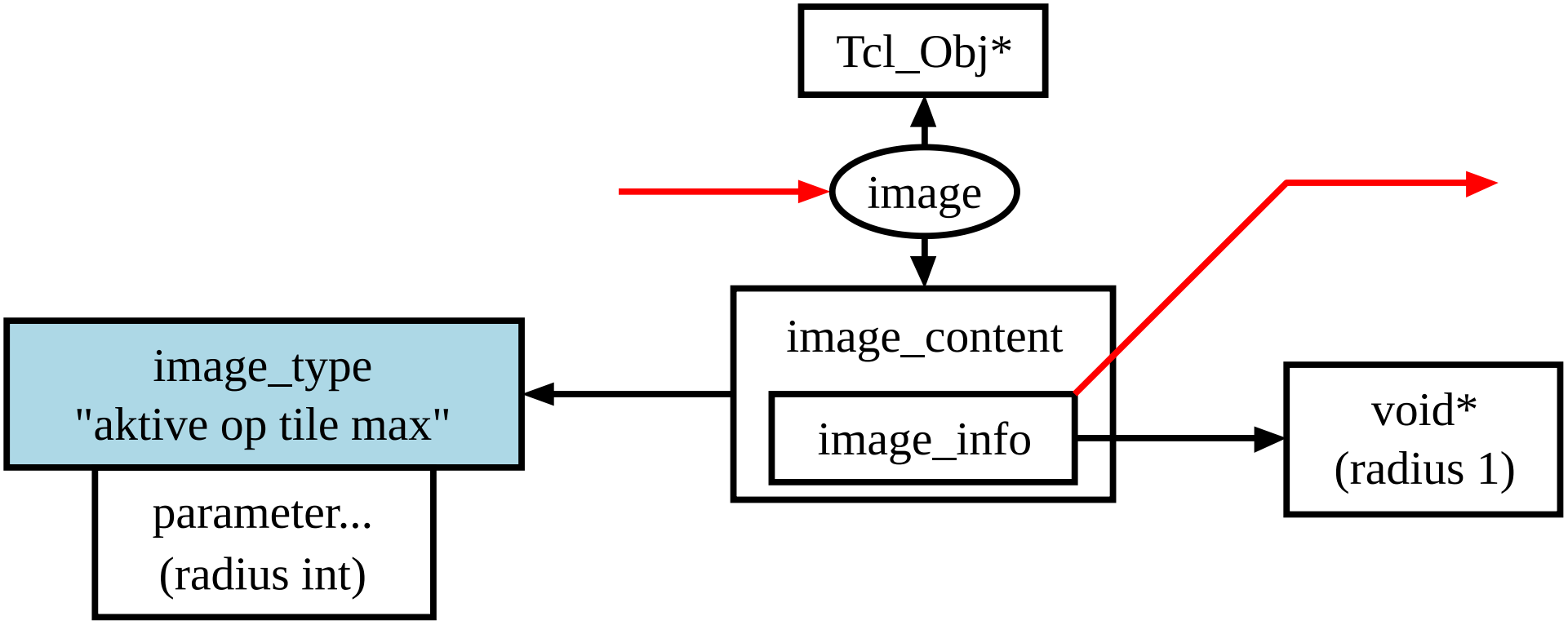
Content and Meta



AK TCL IMAGE VECTOR EXTENSION

DEFINITION STRUCTURES

C Types



AK TCL IMAGE VECTOR EXTENSION

RUNTIME STRUCTURES

- (runtime) state
- (runtime) srcs
- result buffer
- image

AK TCL IMAGE VECTOR EXTENSION

RUNTIME STRUCTURES

- (runtime) state
- (runtime) srcs
- result buffer
- image
  - param
  - type
  - domain
  - state

AK TCL IMAGE VECTOR EXTENSION

RUNTIME STRUCTURES

- **region**: result buffer, image, type
- **region_info**: runtime state, runtime srcs, param, domain, def state

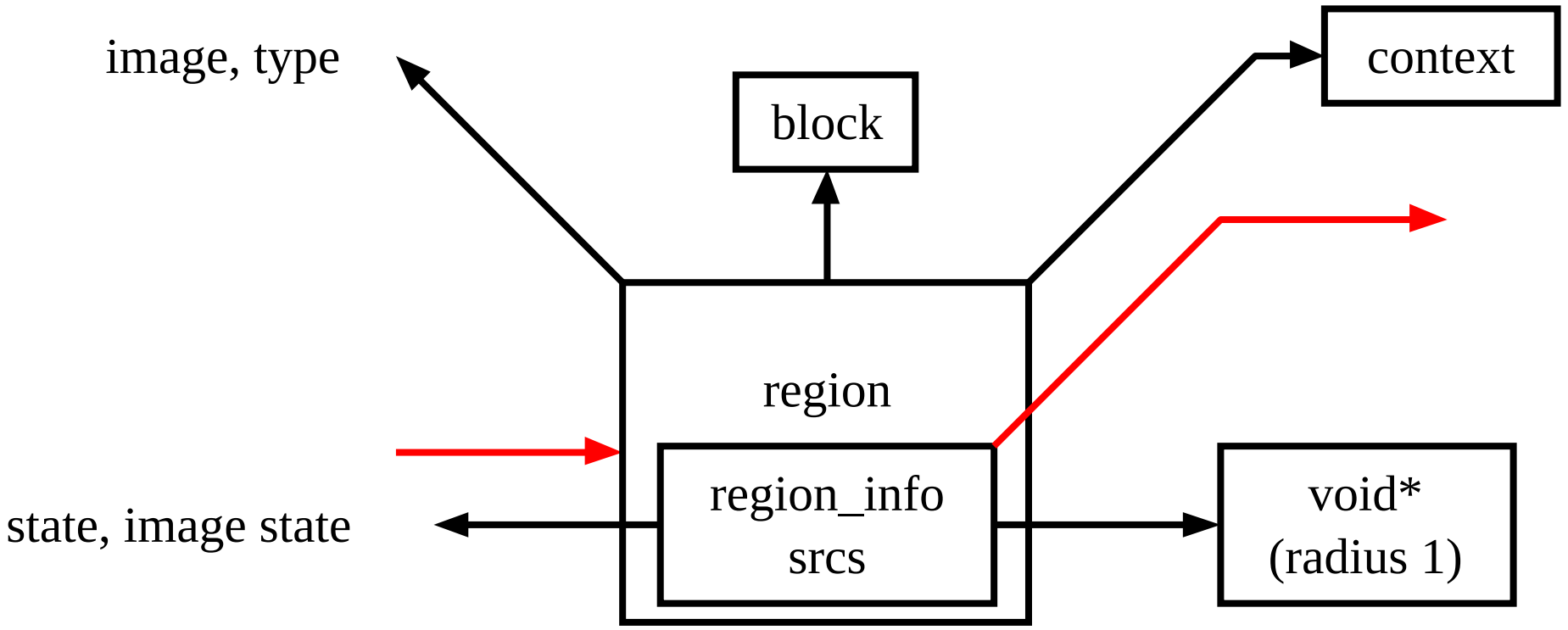
AK TCL IMAGE VECTOR EXTENSION

RUNTIME STRUCTURES

- **region**: result buffer, image, type
- **region_info**: runtime state, runtime srcs, param, domain, def state
- context

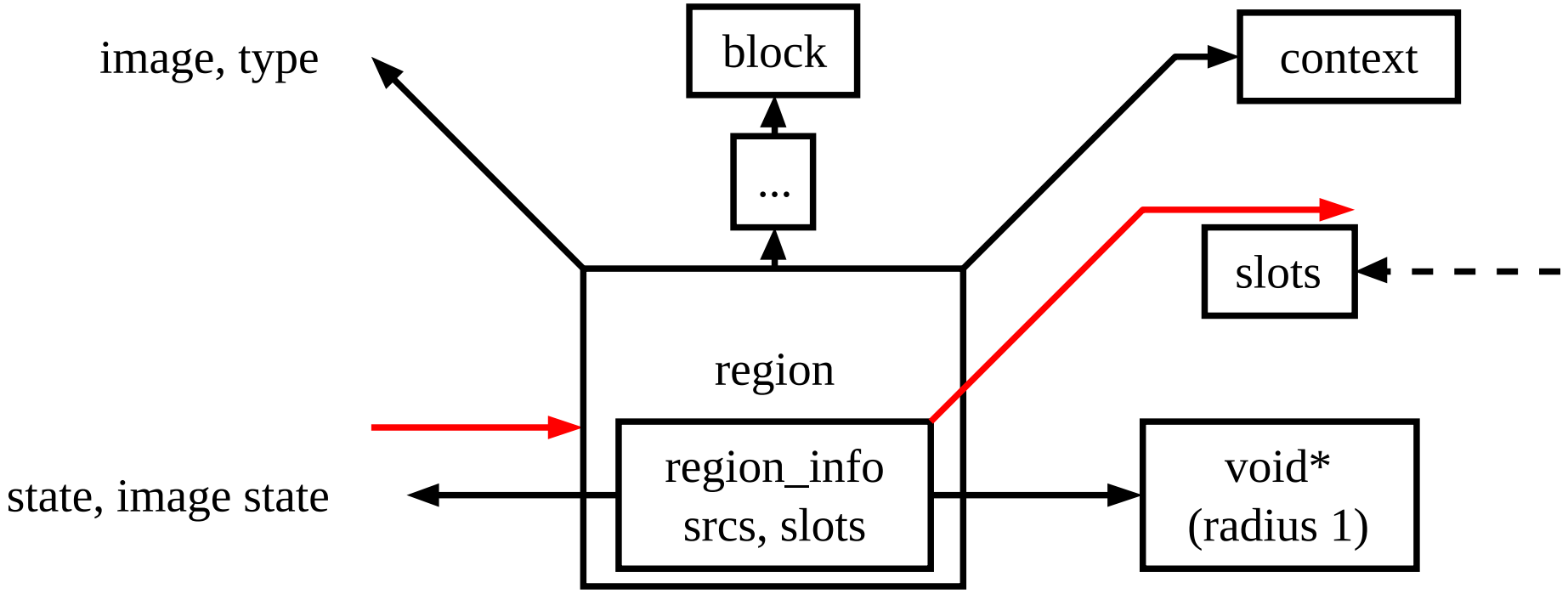
AK TCL IMAGE VECTOR EXTENSION

RUNTIME STRUCTURES



AK TCL IMAGE VECTOR EXTENSION

RUNTIME STRUCTURES



AK TCL IMAGE VECTOR EXTENSION

HORIZONTALLY THREADED EXECUTION

AK TCL IMAGE VECTOR EXTENSION

HORIZONTALLY THREADED EXECUTION

- Generate tasks

AK TCL IMAGE VECTOR EXTENSION

HORIZONTALLY THREADED EXECUTION

- Generate tasks
- Process tasks

AK TCL IMAGE VECTOR EXTENSION

HORIZONTALLY THREADED EXECUTION

- Generate tasks
- Process tasks
- Collect results

AK TCL IMAGE VECTOR EXTENSION

HORIZONTALLY THREADED EXECUTION

- Generate tasks --- **Maker**
- Process tasks
- Collect results

AK TCL IMAGE VECTOR EXTENSION

HORIZONTALLY THREADED EXECUTION

- Generate tasks --- **Maker**
- Process tasks --- **Workers**
- Collect results

AK TCL IMAGE VECTOR EXTENSION

HORIZONTALLY THREADED EXECUTION

- Generate tasks --- **Maker**
- Process tasks --- **Workers**
- Collect results --- **Completer**

AK TCL IMAGE VECTOR EXTENSION

HORIZONTALLY THREADED EXECUTION

- Generate tasks --- **Maker**
- Process tasks --- **Workers**
  - Here are the runtime structures
- Collect results --- **Completer**

AK TCL IMAGE VECTOR EXTENSION

HORIZONTALLY THREADED EXECUTION

- Generate tasks --- **Maker**
- Process tasks --- **Workers**
  - Here are the runtime structures
- Collect results --- **Completer**

Coordination ?

AK TCL IMAGE VECTOR EXTENSION

HORIZONTALLY THREADED EXECUTION

- Generate tasks --- **Maker**
- Process tasks --- **Workers**
  - Here are the runtime structures
- Collect results --- **Completer**

Coordination ?

- Unordered

AK TCL IMAGE VECTOR EXTENSION

HORIZONTALLY THREADED EXECUTION

- Generate tasks --- **Maker**
- Process tasks --- **Workers**
  - Here are the runtime structures
- Collect results --- **Completer**

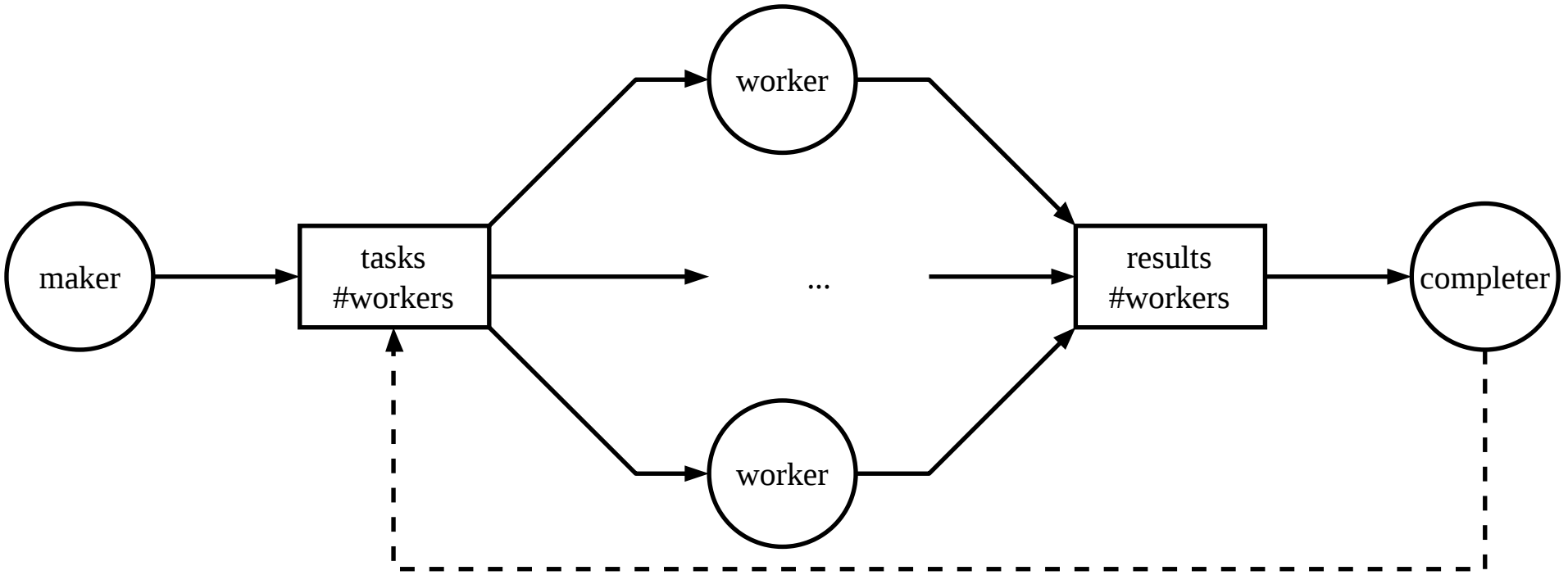
Coordination ?

- Unordered
- Sequential

AK TCL IMAGE VECTOR EXTENSION

HORIZONTALLY THREADED EXECUTION

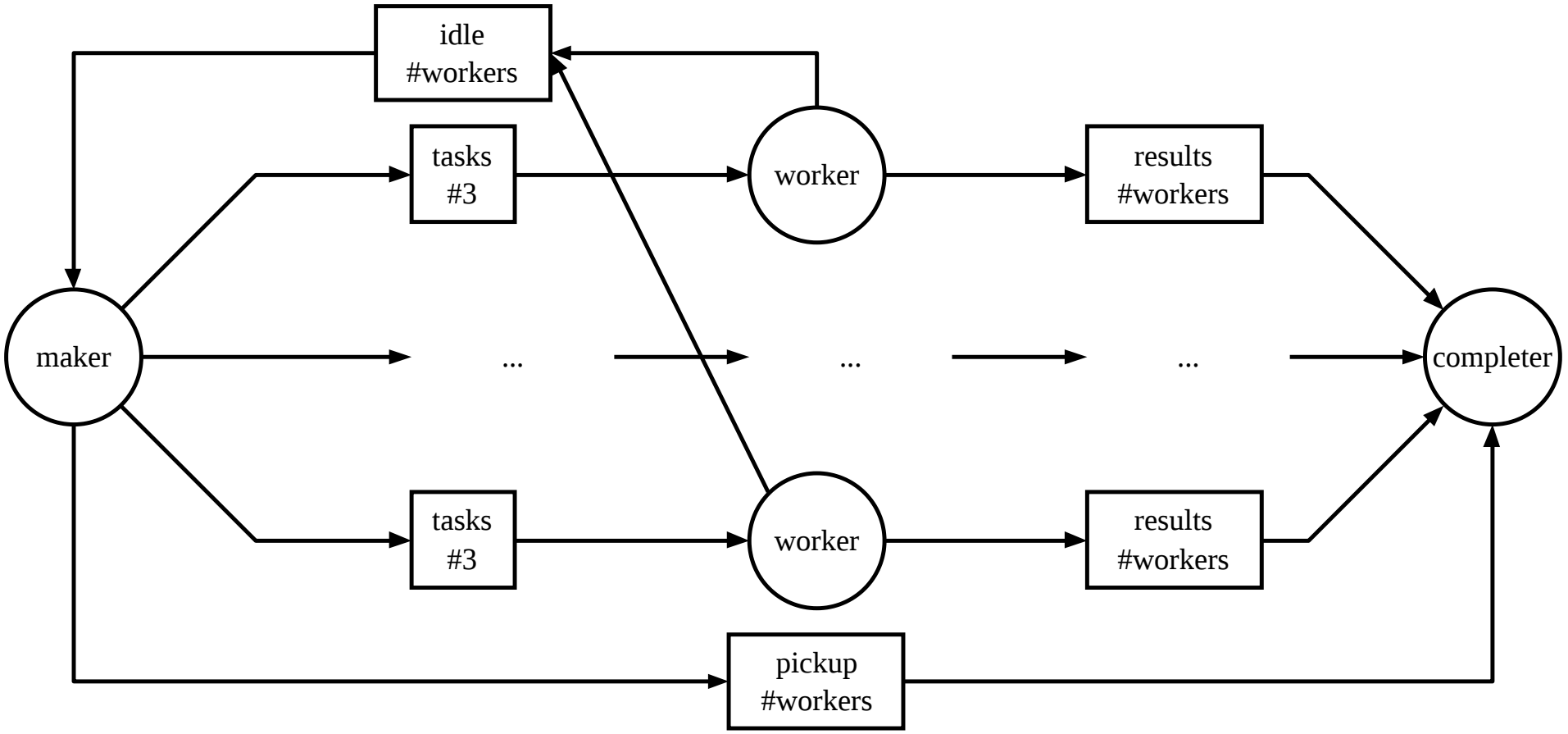
Unordered



AK TCL IMAGE VECTOR EXTENSION

HORIZONTALLY THREADED EXECUTION

Sequential


idle
#workers
tasks
#3
worker
results
#workers
maker
...
...
...
completer
tasks
#3
worker
results
#workers
pickup
#workers

AK TCL IMAGE VECTOR EXTENSION

BENCHMARKING & PERFORMANCE

AK TCL IMAGE VECTOR EXTENSION

BENCHMARKING & PERFORMANCE


```
op-bench {
    aktive image from value // width %W height %H depth 1 value 0.5
}
```

AK TCL IMAGE VECTOR EXTENSION

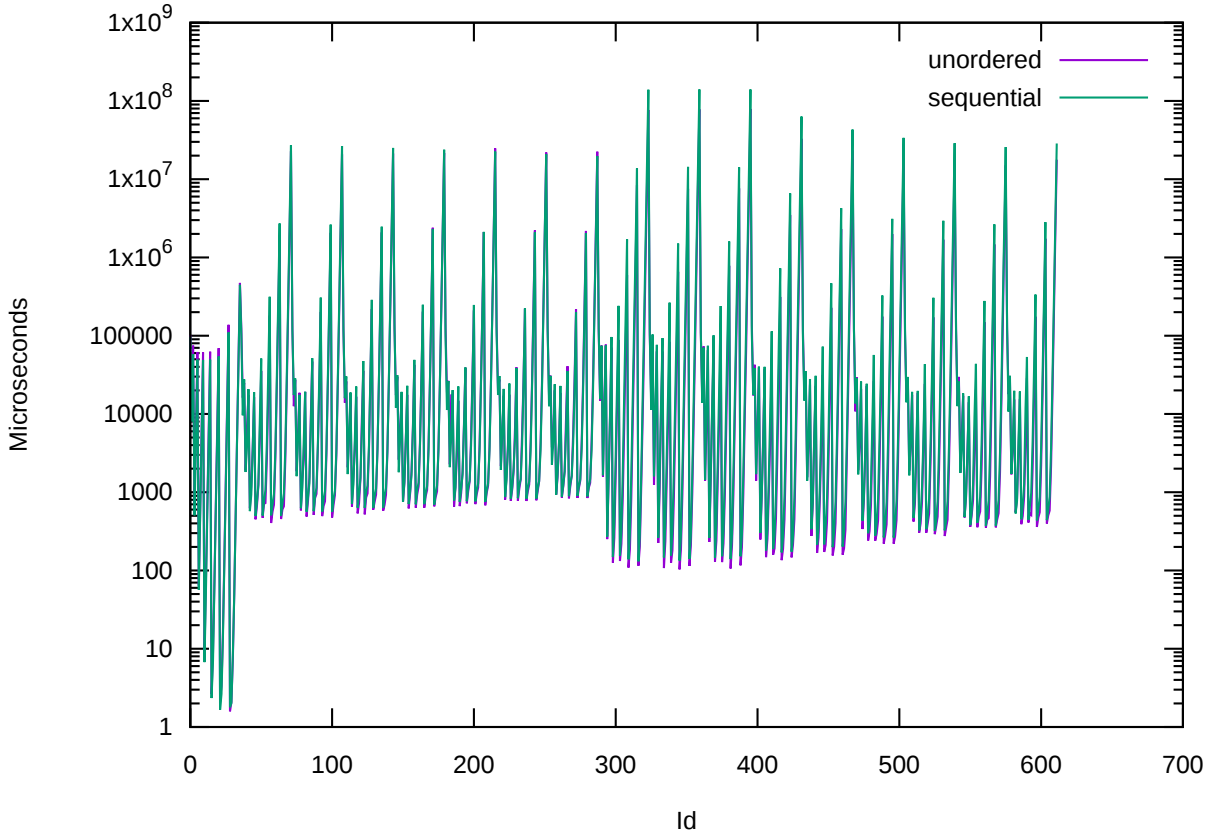
BENCHMARKING & PERFORMANCE


```
    foreach sink {
        null
        null-s
    } {
        # varying concurrency ... unthreaded, ...
        #                     ... threads up to num cpu cores, ...
        #                     ... then overcommitted cpu ...
        foreach cores {
            -1 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
        } {
            # varying sizes ... small to large
            foreach size {
                1 10 100 1000 10000 100000 1000000 10000000
            } { ;#       1K   10K   100K   1M      10M
                # varying shapes of the same size ... tall to wide
                for { set h $size ; set w 1 } { $h > 0 } {
                    set w [expr {$w*10}] ; set h [expr {$h/10}]
                } { ...
```

AK TCL IMAGE VECTOR EXTENSION

BENCHMARKING & PERFORMANCE

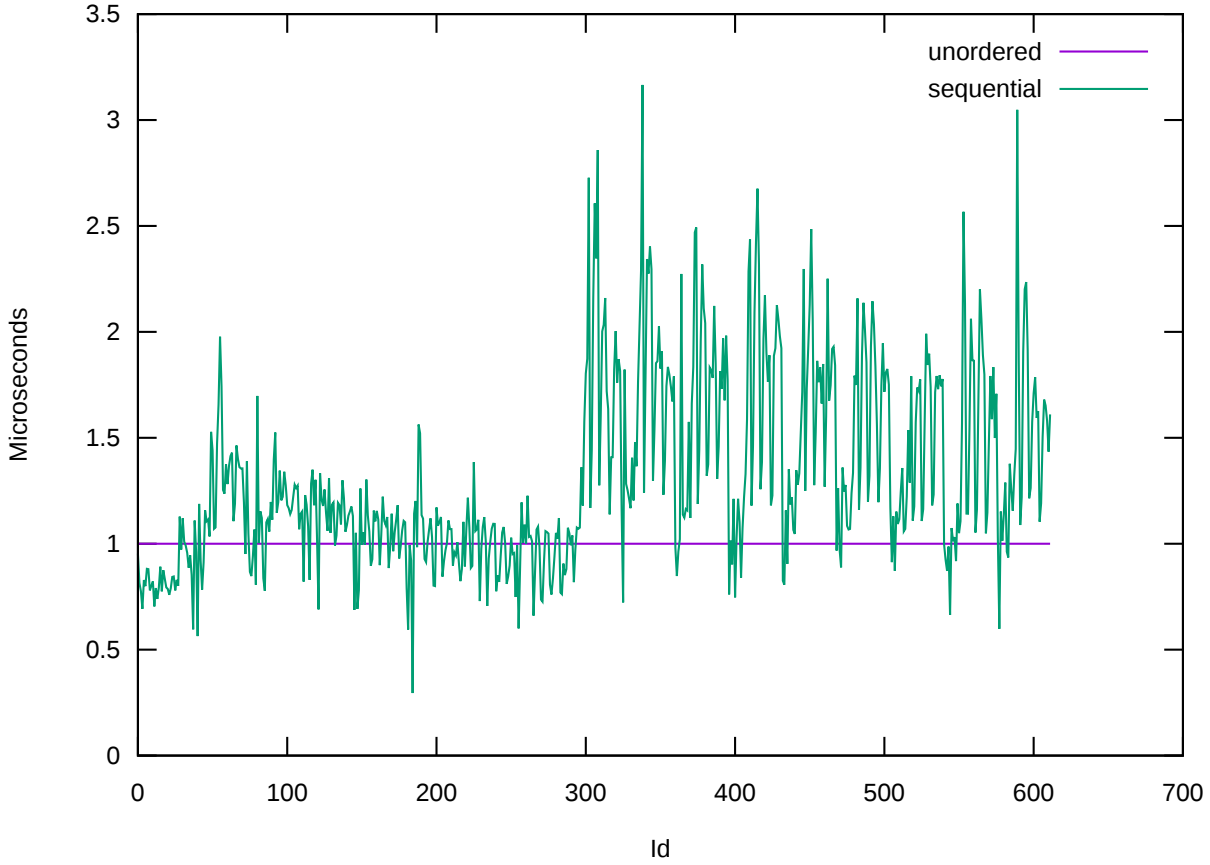
Unordered vs. Sequential


1x10^9
1x10^8
1x10^7
1x10^6
100000
10000
1000
100
10
1
Microseconds
0
100
200
300
400
500
600
700
Id
unordered
sequential

AK TCL IMAGE VECTOR EXTENSION

BENCHMARKING & PERFORMANCE

Unordered vs. Sequential


unordered
sequential
Microseconds
Id
0
0.5
1
1.5
2
2.5
3
3.5
0
100
200
300
400
500
600
700

AK TCL IMAGE VECTOR EXTENSION

BENCHMARKING & PERFORMANCE

- Currently: Performance through threading

AK TCL IMAGE VECTOR EXTENSION

BENCHMARKING & PERFORMANCE

- Currently: Performance through threading
- Can we gain more ? How ?

AK TCL IMAGE VECTOR EXTENSION

BENCHMARKING & PERFORMANCE

- Currently: Performance through threading
- Can we gain more ? How ?
- Make the workers themselves faster!

AK TCL IMAGE VECTOR EXTENSION

BENCHMARKING & PERFORMANCE

- Make the workers themselves faster!

AK TCL IMAGE VECTOR EXTENSION

BENCHMARKING & PERFORMANCE

- Make the workers themselves faster!
  - Idea 1: Loop unrolling

AK TCL IMAGE VECTOR EXTENSION

BENCHMARKING & PERFORMANCE

- Make the workers themselves faster!
  - Idea 1: Loop unrolling
    - Out of Order Superscalar

AK TCL IMAGE VECTOR EXTENSION

BENCHMARKING & PERFORMANCE

- Make the workers themselves faster!
  - Idea 1: Loop unrolling
    - Out of Order Superscalar
  - Idea 2: Vectorization (SIMD)

AK TCL IMAGE VECTOR EXTENSION

BENCHMARKING & PERFORMANCE

- Make the workers themselves faster!
  - Idea 1: Loop unrolling
    - Out of Order Superscalar
  - Idea 2: Vectorization (SIMD)
    - x86 SSE*, AVX*, Arm NEON, SVE, RISC-V "V"

AK TCL IMAGE VECTOR EXTENSION

BENCHMARKING & PERFORMANCE

- Make the workers themselves faster!
  - Idea 1: Loop unrolling
    - Out of Order Superscalar
  - Idea 2: Vectorization (SIMD)
    - x86 SSE*, AVX*, Arm NEON, SVE, RISC-V "V"
    - Google Highway (Portable, Length-Agnostic) @ https://github.com/google/highway

AK TCL IMAGE VECTOR EXTENSION

Function specification


```
set unary0 {
    acos            { scalar { v = acos  (v);     } }
    [...]
}

set unary1 {
    [...]
    shift  {
        scalar  { v = v + a;       }
        highway { v = Add (v, a); }
    }
    [...]
}
```

AK TCL IMAGE VECTOR EXTENSION

Templating


```
set u4_unary0_vdecl {
    extern void aktive_vector4_unary_@name@ (
        double* d, double* s, aktive_uint n);
}
set u4_unary0_vdef {
    void aktive_vector4_unary_@name@ (double* d, double* s, aktive_uint n) {
        // 4-unroll
        for (; n > 3; n -= 4, d += 4, s += 4) {
            double v0 = s[0]; // v1, v2, v3 = s[1,2,3]
#define v v0 // ... v1 v2 v3
            @opcode@
#undef  v
            d[0] = v0; // d[1,2,3] = v1, v2, v3
        }
        // 2-unroll,
        for (; n > 1; n -= 2, d += 2, s += 2) { ... }
        // remainder
        for (; n > 0; n--, d++, s++) { ... }
    }
}
```

AK TCL IMAGE VECTOR EXTENSION

Templating


```
set u1_unary0_hdecl {
    void aktive_highway1_unary_@name@ (
        double* d, double* s, aktive_uint n);}
set u1_unary0_hdef  {
    HWY_ATTR void aktive_highway1_unary_@name@ (
        double* d, double* s, aktive_uint n) {
        const int32_t N = Lanes(f64);
        @decls@
        if (n < N) {    // not a full block - do a masked partial op
            auto mask = FirstN(f64, n);
            auto v    = MaskedLoad(mask, f64, s);
            @opcode@
            BlendedStore (v, mask, f64, d);         return;
        }
        aktive_uint k, border = n - N;  // lane-sized blocks
        for (k = 0; k < n; k += N) {
            aktive_uint at = HWY_MIN (k, border);
            auto v = LoadU (f64, s + at);
            @opcode@
            StoreU (v, f64, d + at);
        }
    }
}
```

AK TCL IMAGE VECTOR EXTENSION

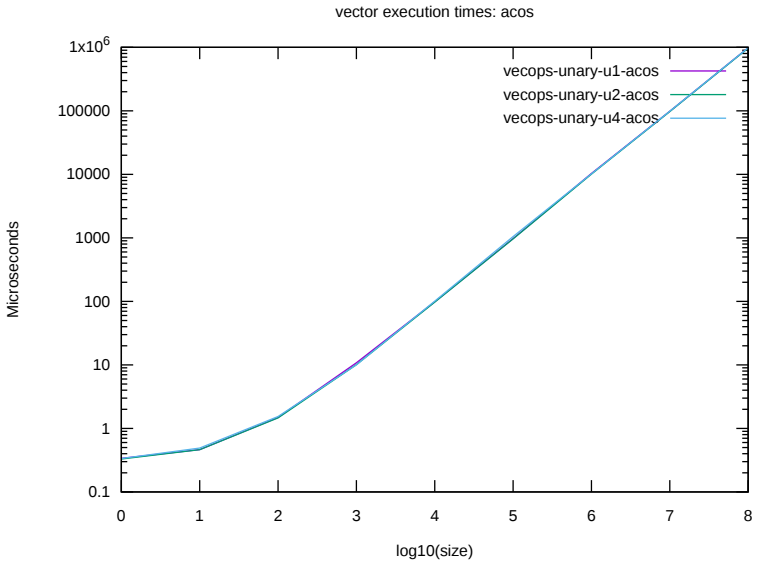
Templating

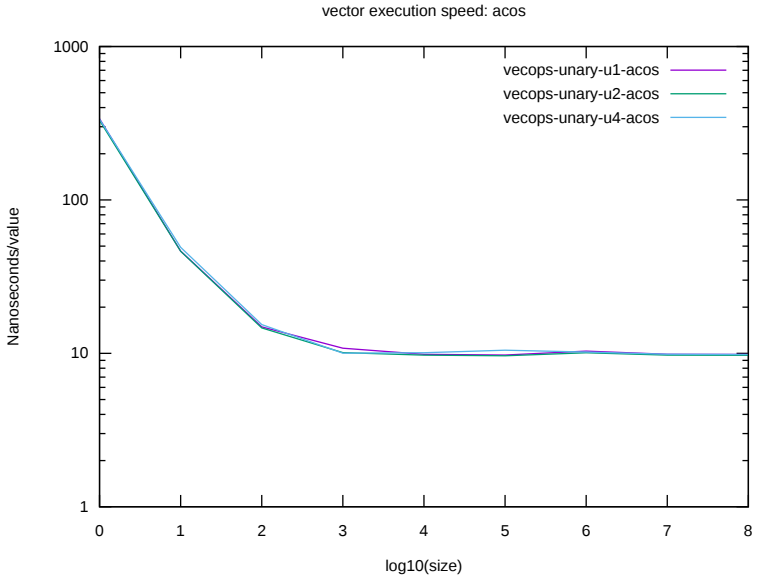

```
set u1_unary0_hdecl {
    void aktive_highway1_unary_@name@ (
        double* d, double* s, aktive_uint n);}
set u1_unary0_hdef  {
    HWY_ATTR void aktive_highway1_unary_@name@ (
        double* d, double* s, aktive_uint n) {
        const int32_t N = Lanes(f64);
        @decls@
        if (n < N) {    // not a full block - do a masked partial op
            auto mask = FirstN(f64, n);
            auto v    = MaskedLoad(mask, f64, s);
            @opcode@
            BlendedStore (v, mask, f64, d);         return;
        }
        aktive_uint k, border = n - N;  // lane-sized blocks
        for (k = 0; k < n; k += N) {
            aktive_uint at = HWY_MIN (k, border);
            auto v = LoadU (f64, s + at);
            @opcode@
            StoreU (v, f64, d + at);
        }
    }
}
```

AK TCL IMAGE VECTOR EXTENSION

BENCHMARKING & PERFORMANCE

Loop Unrolling (1x, 2x, 4x) -- **acos**

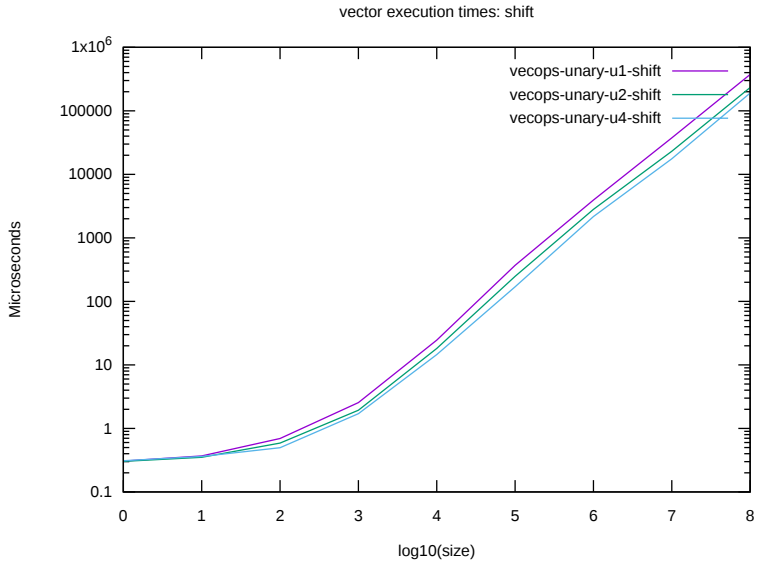


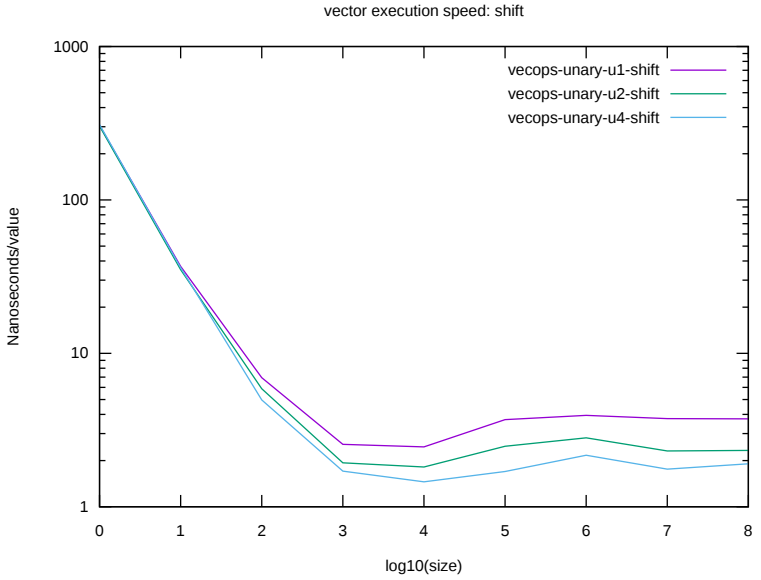


AK TCL IMAGE VECTOR EXTENSION

BENCHMARKING & PERFORMANCE

Loop Unrolling (1x, 2x, 4x) -- **shift**

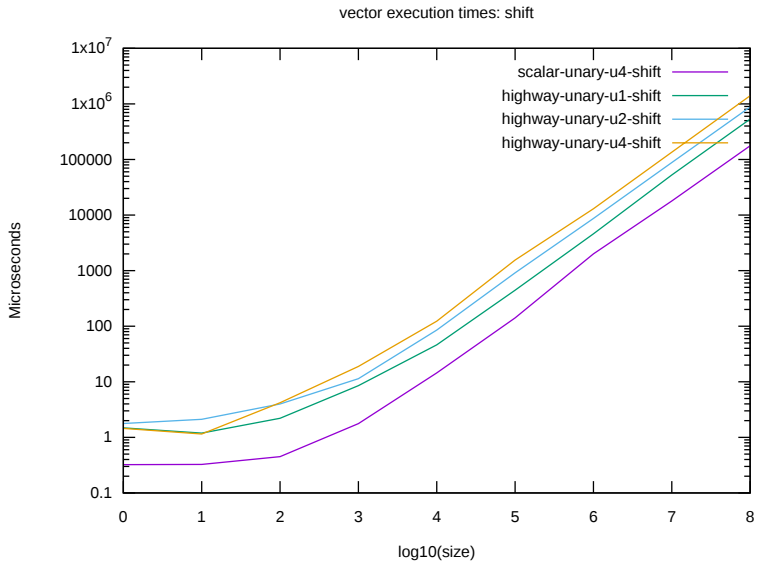


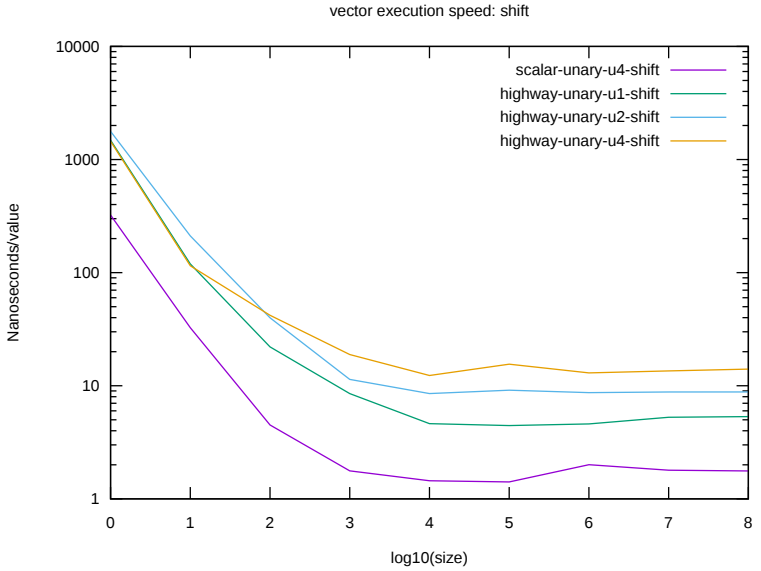


AK TCL IMAGE VECTOR EXTENSION

BENCHMARKING & PERFORMANCE

Loop Unrolling 4x vs Highway -- **shift**





https://core.tcl-lang.org/akupries/aktive/doc/highway-simd-experiment/bench/plots/simd/README.md

https://core.tcl-lang.org/akupries/aktive/timeline?r=highway-simd-experiment

AK TCL IMAGE VECTOR EXTENSION

BENCHMARKING & PERFORMANCE

Benchmarks

https://core.tcl-lang.org/akupries/aktive/doc/trunk/bench/plots/README.md

SIMD Benchmarks

https://core.tcl-lang.org/akupries/aktive/doc/highway-simd-experiment/bench/plots/README.md

AK TCL IMAGE VECTOR EXTENSION

- Questions ? ...