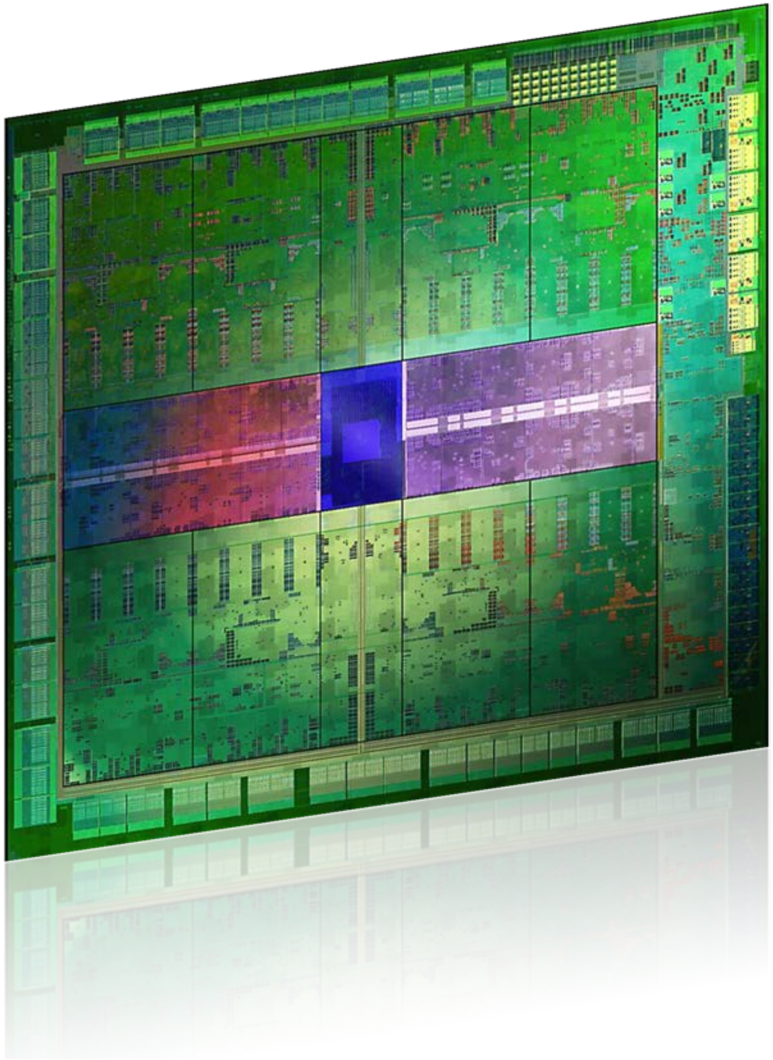
Parallel SWMM

Reducing the runtime of SWMM by parallel computing on commodity Hardware

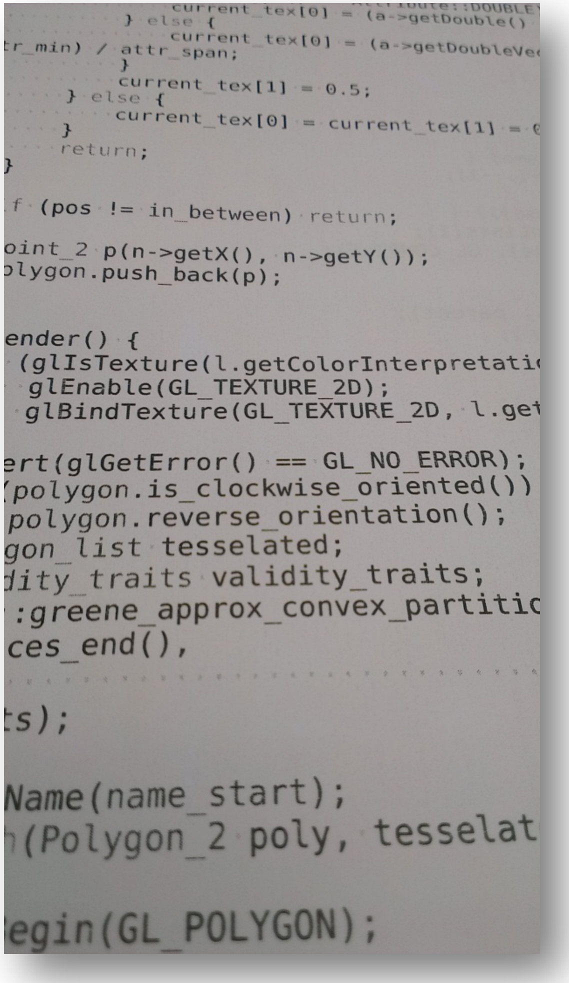
Overview

- **Parallel Computing**
- Method
- Results



Overview

- Parallel Computing
- **Method**
- Results



```
            current_tex[0] = (a->getDouble()
        } else {
            current_tex[0] = (a->getDoubleVe
tr_min) / attr_span;
        }
        current_tex[1] = 0.5;
    } else {
        current_tex[0] = current_tex[1] = 
    }
    return;
}

f (pos != in_between) return;

oint_2 p(n->getX(), n->getY());
olygon.push_back(p);


ender() {
 (glIsTexture(l.getColorInterpretati
  glEnable(GL_TEXTURE_2D);
  glBindTexture(GL_TEXTURE_2D, l.get

ert(glGetError() == GL_NO_ERROR);
(polygon.is_clockwise_oriented())
 polygon.reverse_orientation();
gon_list tesselated;
dity_traits validity_traits;
 :greene_approx_convex_partitic
 ces_end(),

ts);


Name(name_start);
h(Polygon_2 poly, tesselat


egin(GL_POLYGON);
```

Overview

- Parallel Computing
- Method
- **Results**

Multicore Revolution

- **“The free lunch … is over”**

- Parallel Computing

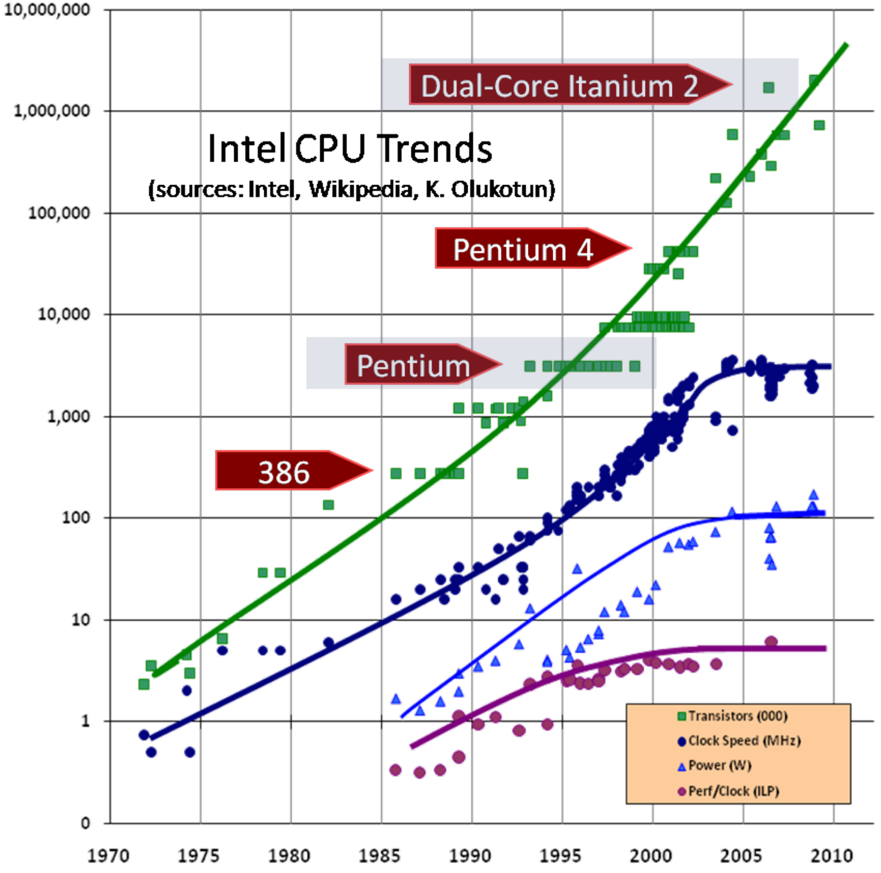
- Challenges

Multicore Revolution

- **“The free lunch … is over”**

- Parallel Computing

- Challenges



Multicore Revolution

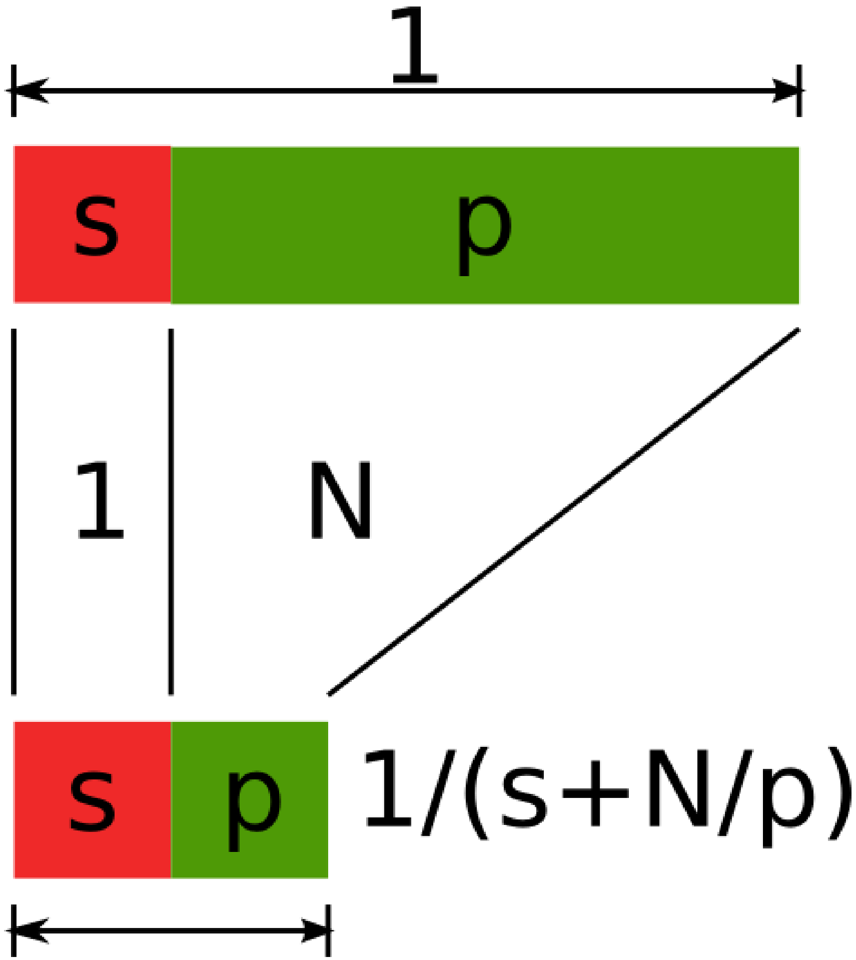
- “The free lunch … is over”

- **Parallel Computing**

- Challenges

Multicore Revolution

- “The free lunch … is over”
- **Parallel Computing**
- Challenges



Multicore Revolution

- “The free lunch … is over”

- Parallel Computing

- **Challenges**

Multicore Revolution

- "The free lunch … is over"
- Parallel Computing
- **Challenges**

Parallel Computing …

- It is hard…
- It is new…

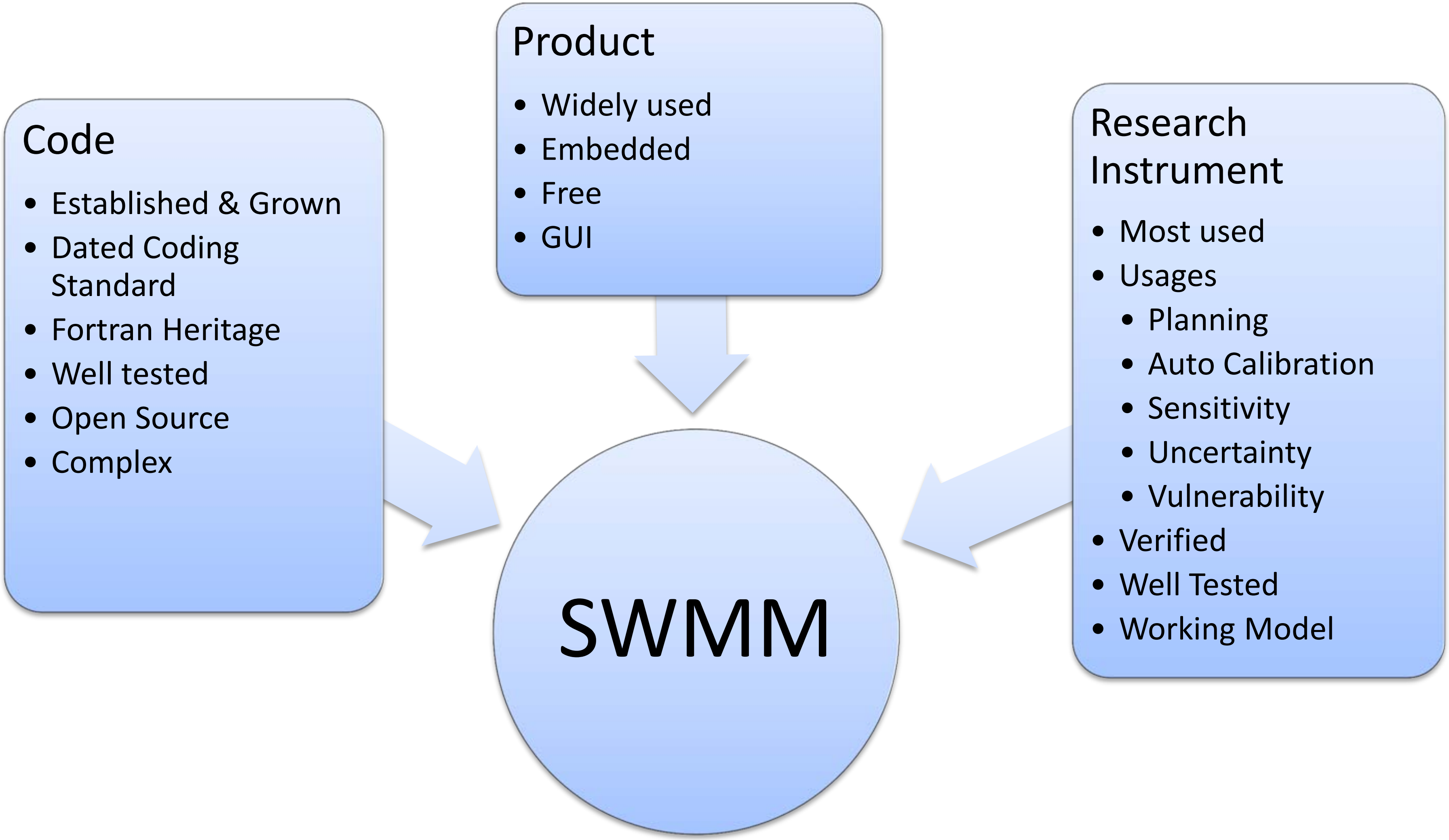
But we need to do it!

Parallel Computing …

- It is hard…
- It is new…

But we need to do it!

Views of SWMM



Code

- Established & Grown
- Dated Coding Standard
- Fortran Heritage
- Well tested
- Open Source
- Complex

Product

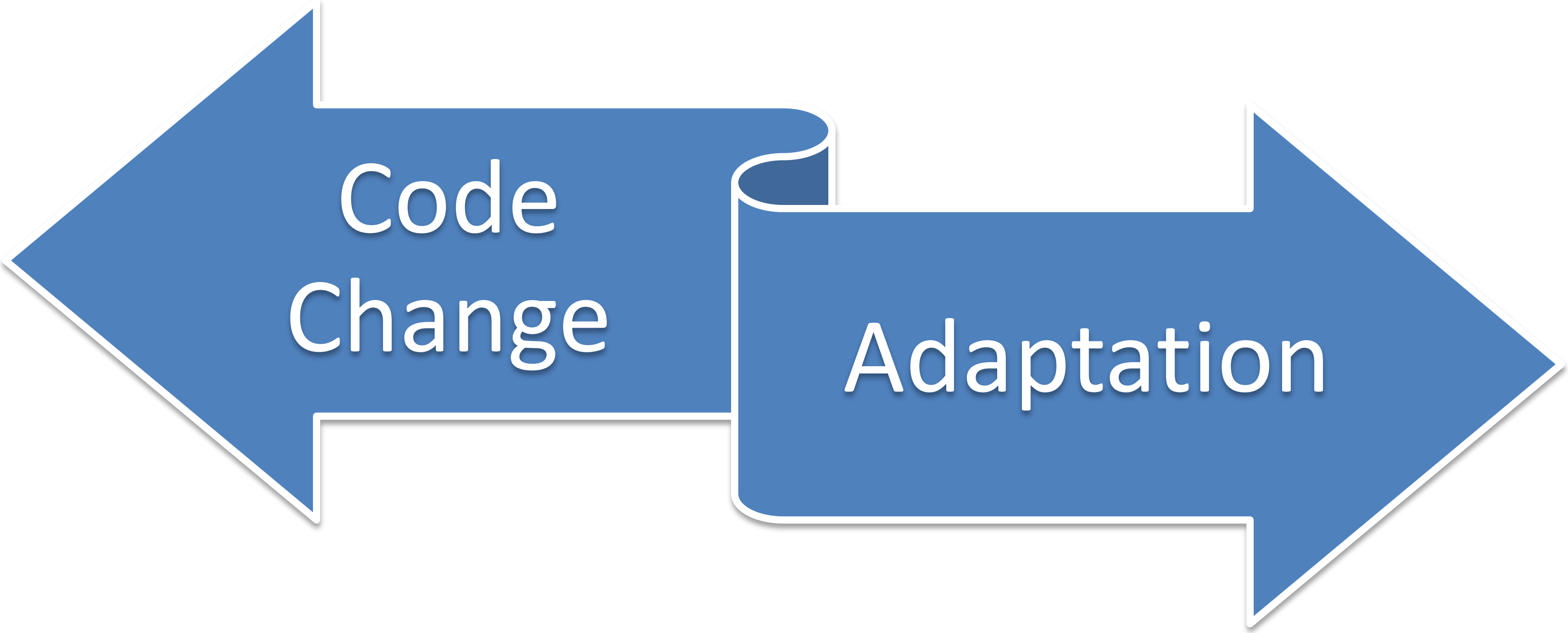
- Widely used
- Embedded
- Free
- GUI

Research Instrument

- Most used
- Usages
  - Planning
  - Auto Calibration
  - Sensitivity
  - Uncertainty
  - Vulnerability
- Verified
- Well Tested
- Working Model

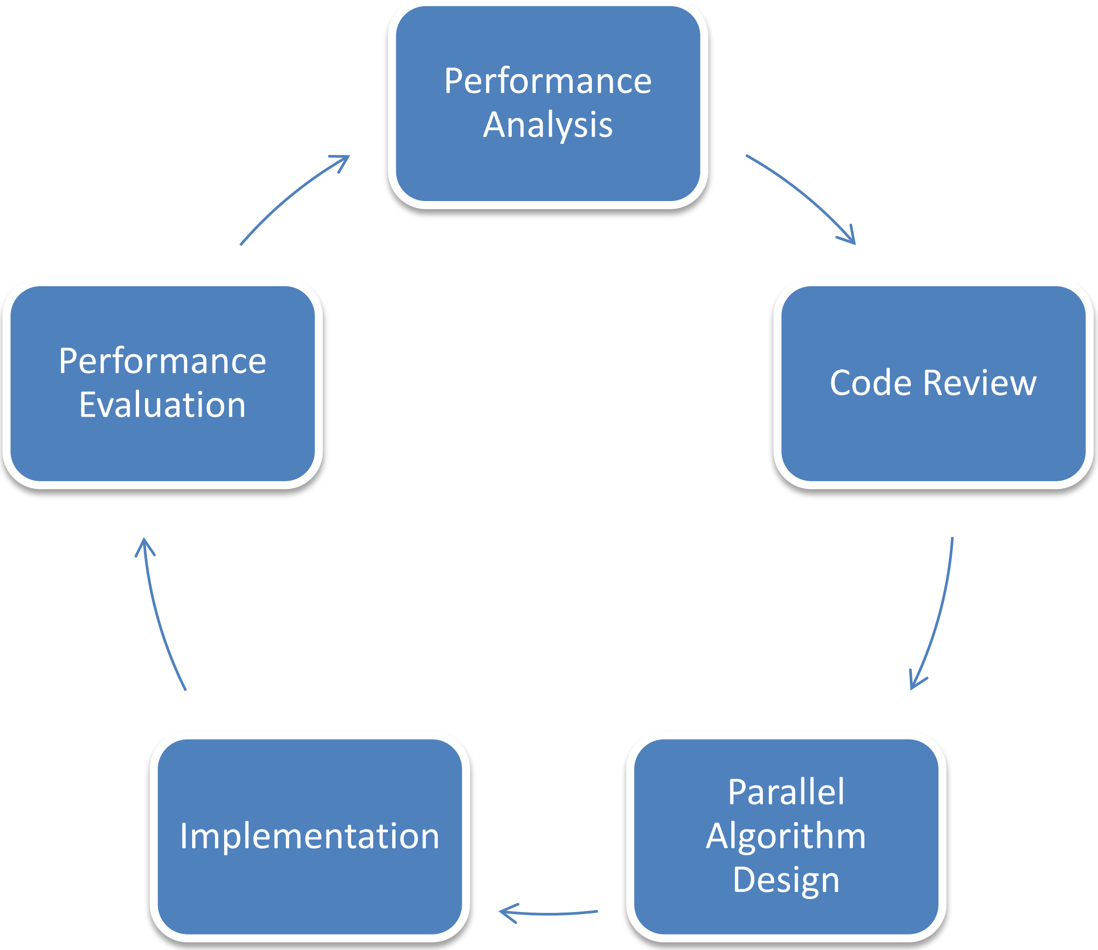
SWMM

The Crux


Code Change

Adaptation

Strategy



Performance Analysis

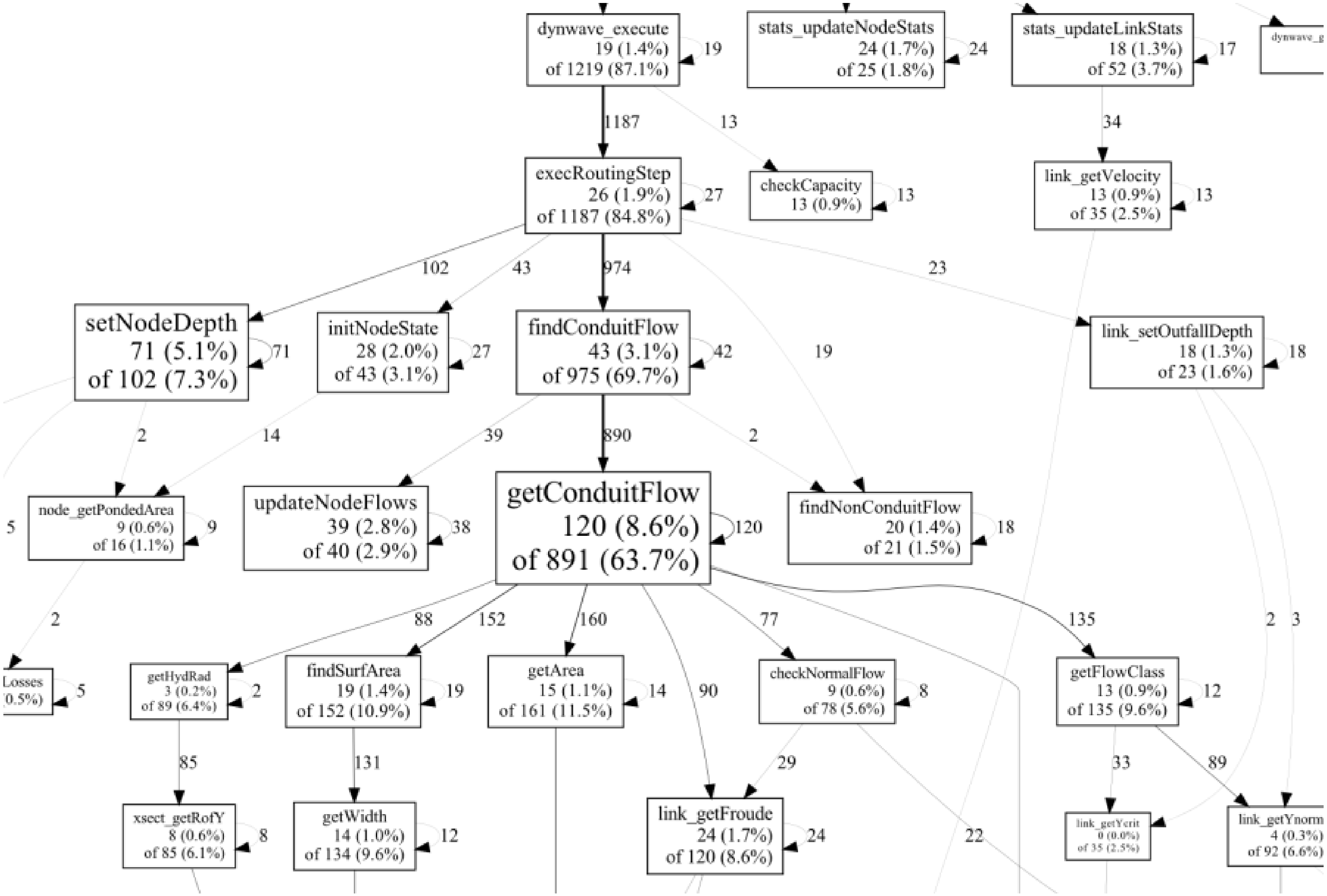
Code Review

Parallel Algorithm Design

Implementation

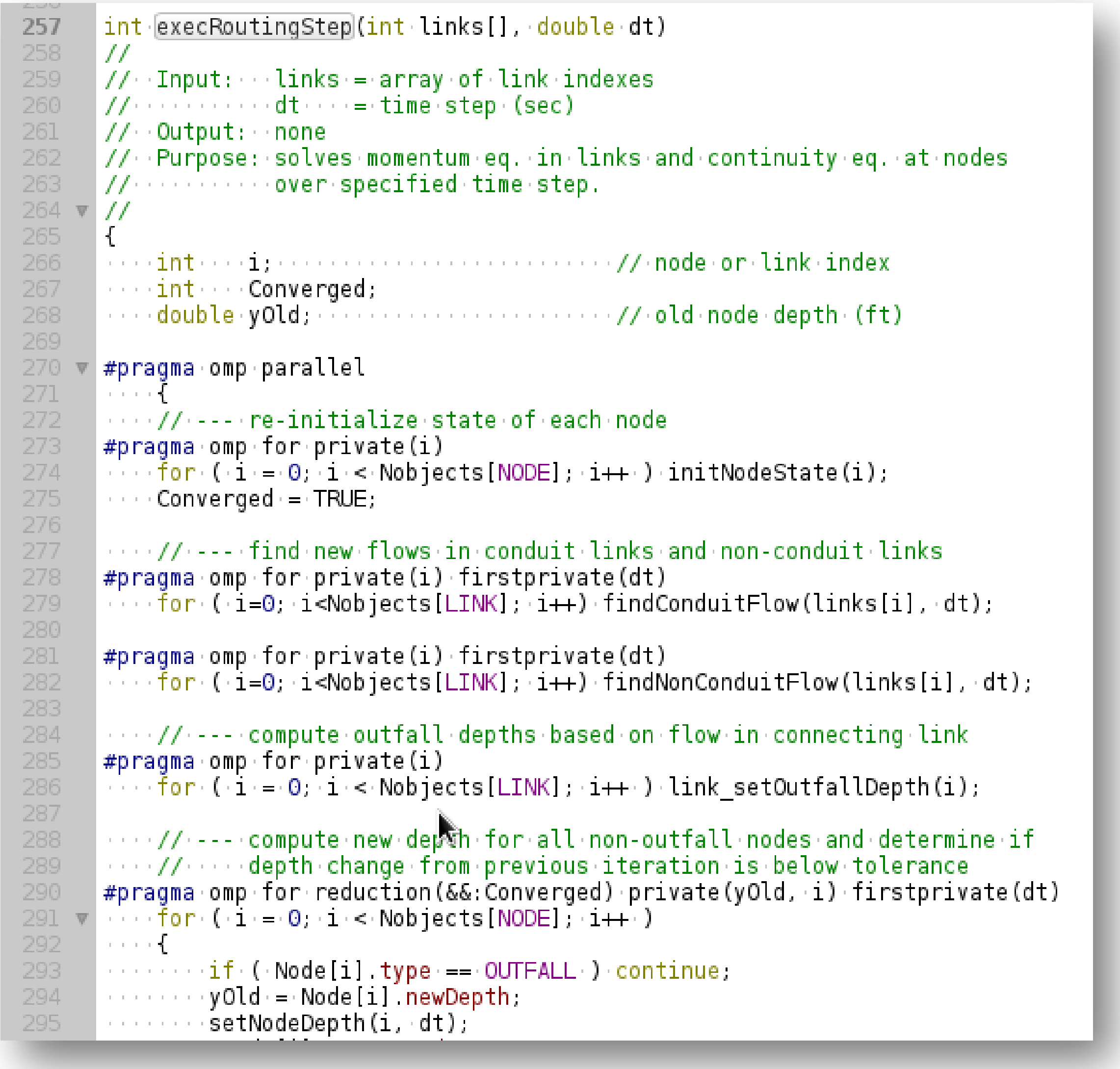
Performance Evaluation

Performance Analysis



dynwave_execute 19 (1.4%) of 1219 (87.1%)
19
stats_updateNodeStats 24 (1.7%) of 25 (1.8%)
24
stats_updateLinkStats 18 (1.3%) of 52 (3.7%)
17
dynwave_g
1187
13
34
execRoutingStep 26 (1.9%) of 1187 (84.8%)
27
checkCapacity 13 (0.9%)
13
link_getVelocity 13 (0.9%) of 35 (2.5%)
13
102
43
974
23
setNodeDepth 71 (5.1%) of 102 (7.3%)
71
initNodeState 28 (2.0%) of 43 (3.1%)
27
findConduitFlow 43 (3.1%) of 975 (69.7%)
42
19
link_setOutfallDepth 18 (1.3%) of 23 (1.6%)
18
2
14
39
890
2
5
node_getPondedArea 9 (0.6%) of 16 (1.1%)
9
updateNodeFlows 39 (2.8%) of 40 (2.9%)
38
getConduitFlow 120 (8.6%) of 891 (63.7%)
120
findNonConduitFlow 20 (1.4%) of 21 (1.5%)
18
2
88
152
160
77
135
2
3
Losses (0.5%)
5
getHydRad 3 (0.2%) of 89 (6.4%)
2
findSurfArea 19 (1.4%) of 152 (10.9%)
19
getArea 15 (1.1%) of 161 (11.5%)
14
90
checkNormalFlow 9 (0.6%) of 78 (5.6%)
8
getFlowClass 13 (0.9%) of 135 (9.6%)
12
85
131
29
33
89
xsect_getRofY 8 (0.6%) of 85 (6.1%)
8
getWidth 14 (1.0%) of 134 (9.6%)
12
link_getFroude 24 (1.7%) of 120 (8.6%)
24
22
link_getYcrit 0 (0.0%) of 35 (2.5%)
link_getYnorm 4 (0.3%) of 92 (6.6%)

OpenMP Implementation



```
int execRoutingStep(int links[], double dt)
//
//  Input:   links = array of link indexes
//           dt    = time step (sec)
//  Output:  none
//  Purpose: solves momentum eq. in links and continuity eq. at nodes
//           over specified time step.
//
{
    int    i;                          // node or link index
    int    Converged;
    double yOld;                       // old node depth (ft)

#pragma omp parallel
    {
    // --- re-initialize state of each node
#pragma omp for private(i)
    for ( i = 0; i < Nobjects[NODE]; i++ ) initNodeState(i);
    Converged = TRUE;

    // --- find new flows in conduit links and non-conduit links
#pragma omp for private(i) firstprivate(dt)
    for ( i=0; i<Nobjects[LINK]; i++) findConduitFlow(links[i], dt);

#pragma omp for private(i) firstprivate(dt)
    for ( i=0; i<Nobjects[LINK]; i++) findNonConduitFlow(links[i], dt);

    // --- compute outfall depths based on flow in connecting link
#pragma omp for private(i)
    for ( i = 0; i < Nobjects[LINK]; i++ ) link_setOutfallDepth(i);

    // --- compute new depth for all non-outfall nodes and determine if
    //     depth change from previous iteration is below tolerance
#pragma omp for reduction(&&:Converged) private(yOld, i) firstprivate(dt)
    for ( i = 0; i < Nobjects[NODE]; i++ )
    {
        if ( Node[i].type == OUTFALL ) continue;
        yOld = Node[i].newDepth;
        setNodeDepth(i, dt);
```

Performance Evaluation

Hardware

- Dual Socket XEON X5650 @ 2.67 GHz
- 6 Cores/Socket => 24 Threads
- 24 GB RAM

Classic Benchmarking

- AVG of four
- Runs for 1, 2, 4, 6, …, 24 Threads
- Hydraulics only

Method

- Widely used product / research instrument

- Low impact implementation in a grown code

- Needs refined Software Management

Method

- Widely used product / research instrument

- Low impact implementation in a grown code

- Needs refined Software Management

Method

- Widely used product / research instrument

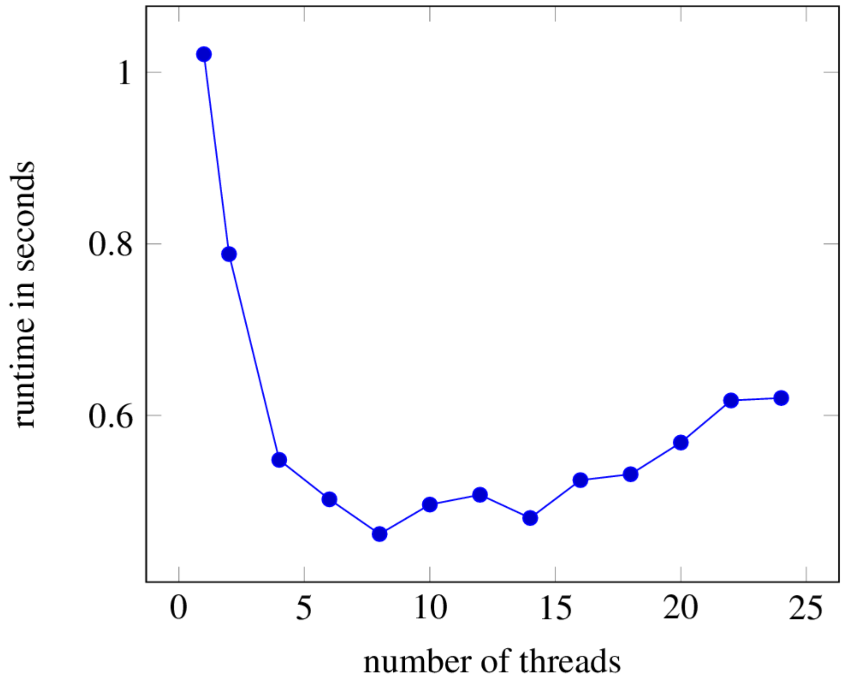
- Low impact implementation in a grown code

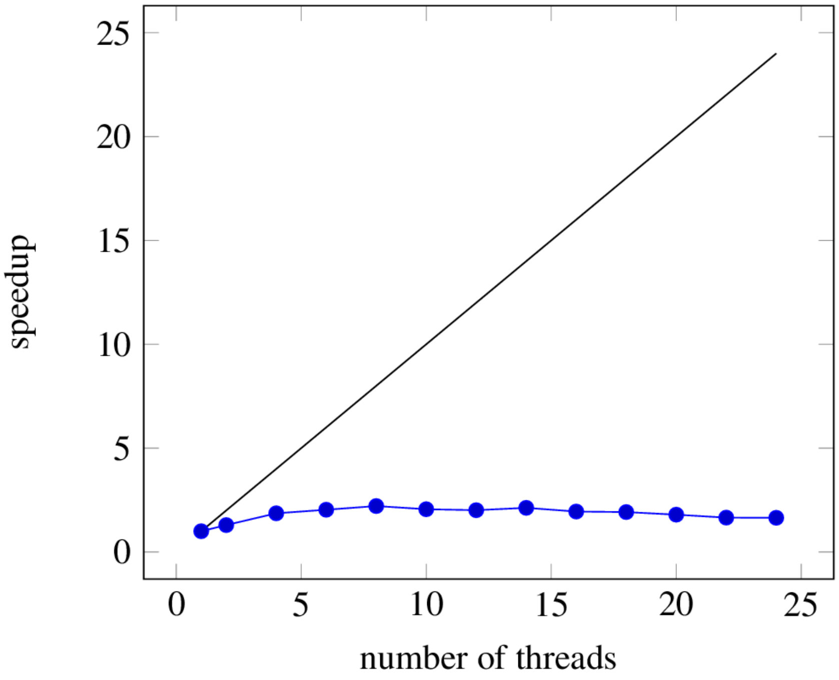
- Needs refined Software Management

Input Systems

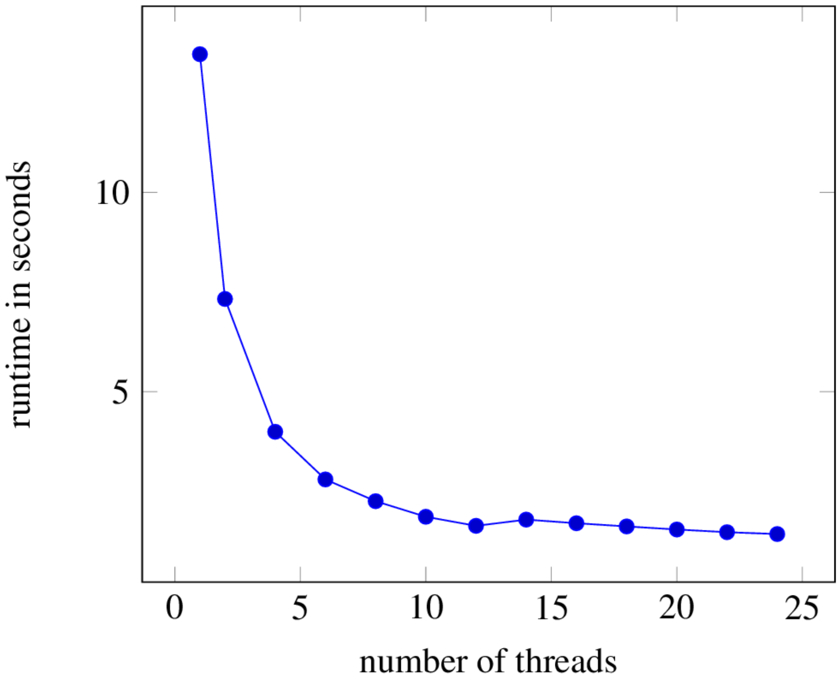
Input System	# Nodes	# Links	# Catchments	Population
Artificial	50	49	42	Unknown
Village	1709	1722	440	10760
Small Town	1254	1274	3062	12695
Town	5485	5834	4498	120147

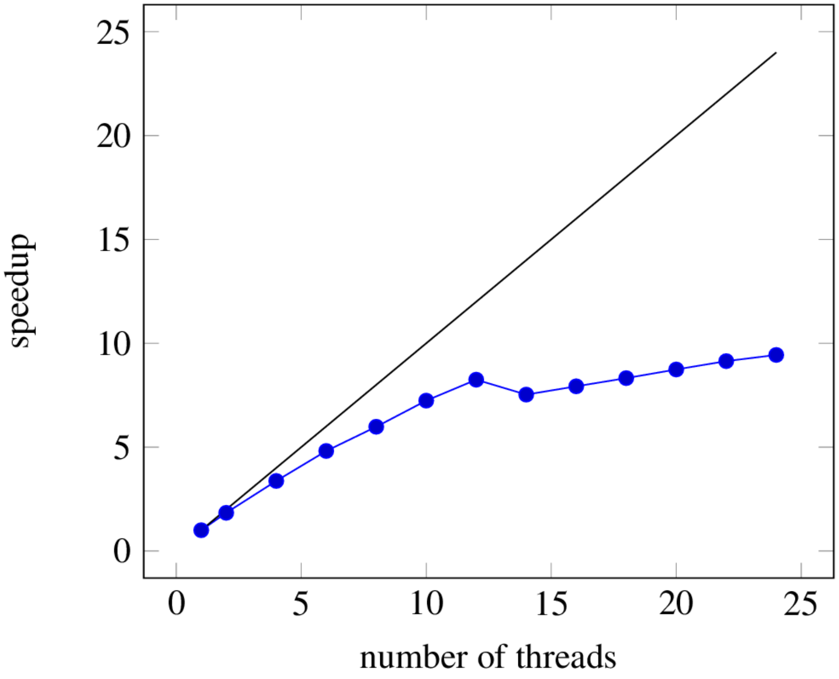
Results – CSG


runtime in seconds
1
0.8
0.6
0
5
10
15
20
25
number of threads

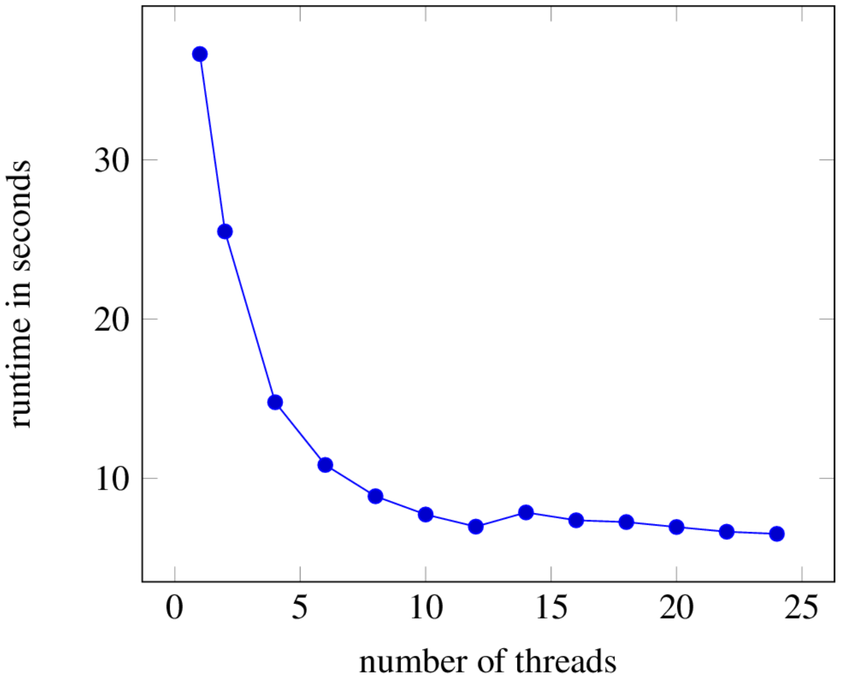

speedup
25
20
15
10
5
0
0
5
10
15
20
25
number of threads

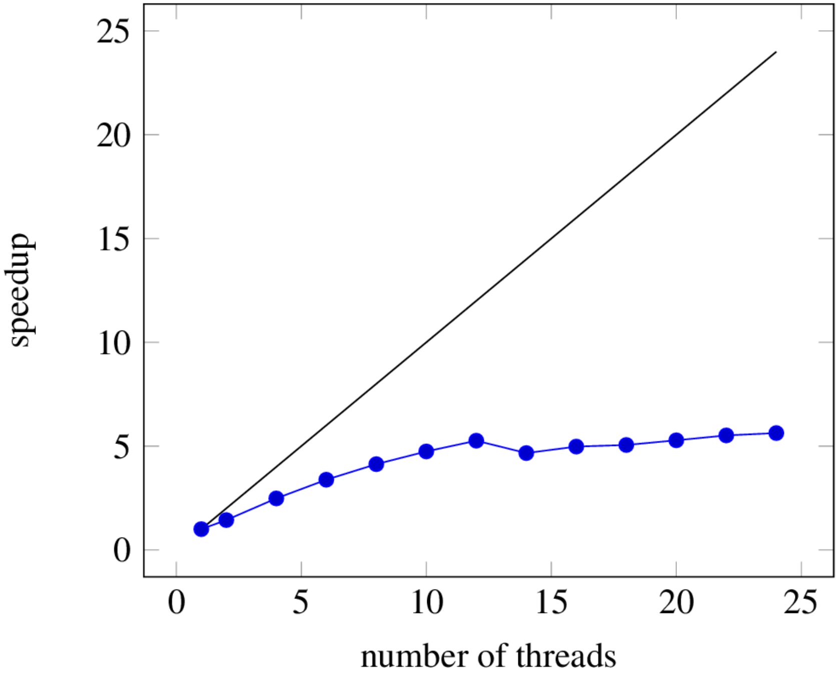
Results – Village



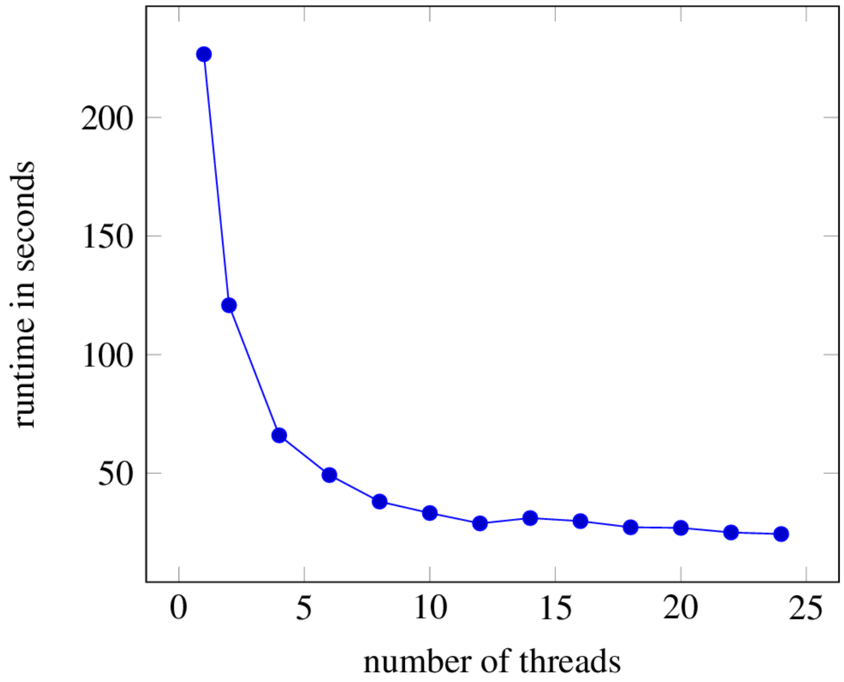


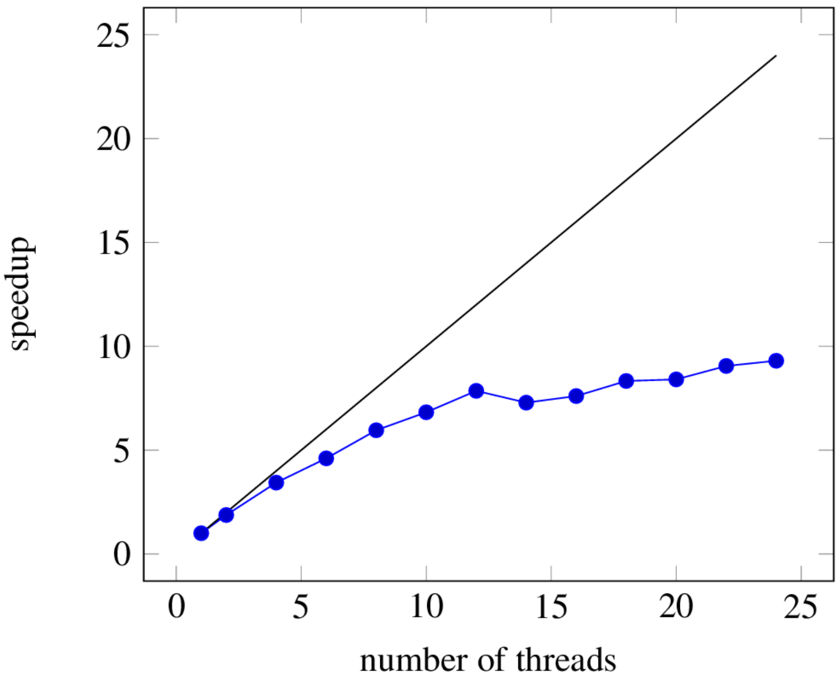
Results – Small Town





Results - Town





Conclusion and Outlook

- 9.3 Speedup
- No overhead Introduced
- Minimal code changes

- Look at other parts
- GPU implementation

Conclusion and Outlook

- 9.3 Speedup
- No overhead Introduced
- Minimal code changes

- Look at other parts
- GPU implementation

THANK YOU
FOR LISTENING!