INTEL® HPC DEVELOPER CONFERENCE
FUEL YOUR INSIGHT
Reshaping core genomics software tools for the many-core era

Ben Langmead
Assistant Professor
Johns Hopkins University, Department of Computer Science
November 2016
The Intel Parallel Computing Center at

Ben Langmead
Valentin Antonescu
Chris Wilks
Steven Salzberg
Daehwan Kim
Outline

- Introduction to sequencing data analysis & Bowtie
- Thread scaling improvements using TBB
  - Choice of mutex
  - Two-stage parsing
- AVX2, AVX512-KNC & AVX512-KNC improvements
- Impact on the field
Sequencing
Sequencing
Sequencing

1 Pbp

Terabases

SRA database growth

Total

Open access

3 -> 6 Pbp in ~18 months
Read alignment
Read alignment

- Needle in a haystack
- Billions of reads from a single week-long sequencing run
- Human reference genome is ~3B bases (letters) long
Bowtie and Bowtie 2

- Together cited by >12K other scientific studies since 2009
- Bundled with dozens of other tools & many Linux distros
HISAT

- Based on Bowtie 2 and a leading *spliced* aligner for RNA sequencing data
- Cited in >75 scientific studies since 2015
Design of Bowtie & Bowtie 2

Bowtie 1

Bowtie 2
Design of Bowtie & Bowtie 2

Bowtie 1

Read → Alignment → Hit

Random access to large index data structure and minimal ILP

Bowtie 2

Read → Read substring → Ref substring → Alignment

INTEL® HPC DEVELOPER CONFERENCE
Design of Bowtie & Bowtie 2

Dynamic programming, lots of ILP
Thread scaling

Vertical axis is per-thread running time; lower is better

- Switching to analogous TBB lock could bring big improvement
Thread scaling

- Removing synchronization by “stubbing” input lock gives further improvement
Thread scaling

- Vtune investigation indicates synchronization itself (e.g. see __TBB_LockByte) is taking the time
Thread scaling

**Bowtie 2 unpaired**

How to close the gap between actual and ideal performance?
Thread scaling

Why does mutex choice have outsize effect?
Thread scaling

- Mutex spinning on atomic op (compare-and-swap, test-and-set), spurs exchange of cache coherence messages
- Image by Kayvon Fatahalian, Copyright 2015 Carnegie Mellon University
Thread scaling

- Even a standard pthreads mutex was outperforming the spin lock when running one thread per available core
  - More evidence that cache coherence traffic is culprit

- Queue locks are known to have better cache properties
  - Waiting thread spins on normal (non-atomic) read
  - Cache line read belongs exclusively to that thread and can live in L1
Thread scaling

- We hypothesized a NUMA-aware “cohort lock” could help further

Cohort locking

class CohortLock {
public:
    CohortLock() : lockers_numa_idx(-1) {
        starvation_counters = new int[MAX_NODES]();
        own_global = new bool[MAX_NODES]();
        local_locks = new TKTLock[MAX_NODES];
    }
    ~CohortLock() {
        delete[] starvation_counters;
        delete[] own_global;
        delete[] local_locks;
    }
    void lock();
    void unlock();
private:
    static const int STARVATION_LIMIT = 100;
    static const int MAX_NODES = 128;
    volatile int* starvation_counters; // 1 per node
    volatile bool* own_global; // 1 per node
    volatile int lockers_numa_idx; // 1 per node
    TKTLock* local_locks; // 1 per node
    PTLLock global_lock;
};

- Each NUMA node has per-node ticket lock
- Other per-node information tracks when to pass lock to other threads on same node
- Single global partitioned ticket lock
Cohort locking

void CohortLock::lock() {
    const int numa_idx = determine_numa_idx();
    local_locks[numa_idx].lock();
    if(!own_global[numa_idx]) {
        global_lock.lock();
    }
    starvation_counters[numa_idx]++;
    own_global[numa_idx] = true;
    lockers_numa_idx = numa_idx;
}

void CohortLock::unlock() {
    assert(lockers_numa_idx != -1);
    int numa_idx = lockers_numa_idx;
    lockers_numa_idx = -1;
    if(local_locks[numa_idx].q_length() == 1 ||
        starvation_counters[numa_idx] > STARVATION_LIMIT)
    {
        global_lock.unlock();
        starvation_counters[numa_idx] = 0;
        own_global[numa_idx] = false;
    }
    local_locks[numa_idx].unlock();
}

- When locking:
  - Grab local lock
  - Once grabbed, grab global lock if not already owned by this node

- When unlocking:
  - Is another thread on same node queued? If so, hand lock to next in queue
  - Otherwise release global & local locks
  - Override hand-off if others are starving
Cohort locking

- Another implementation of cohort locking available in ConcurrencyKit: http://concurrencykit.org
  - https://github.com/concurrencykit/ck/blob/master/include/ck_cohort.h
Thread scaling

- Chris Wilks added TBB queue locks, JHU/TBB Cohort locks (2 flavors) to Bowtie 2, Bowtie & HISAT
- Available in public branches, with all but cohort locks available in master branch and in recent releases
Thread scaling

- Novel strategy splits input parsing into two “phases”
- First (“light parsing”) rapidly detects record boundaries, requiring synchronization but with very brief critical section
- Second (“full parsing”) fully parses each record (pictured, right) with no synchronization
- Minimizes time spent in crucial critical section
Thread scaling: Bowtie 2 unpaired

- TBB queuing_mutex and TBB/JHU cohort lock perform best
Thread scaling: Bowtie 2 unpaired

- Two-phase parsing yields substantial thread-scaling boost; close to perfect up to 120 threads, regardless of mutex
Thread scaling: Bowtie 2 paired-end

- queuing_mutex and cohort lock again perform the best, near ideal
Thread scaling: Bowtie 2 paired-end

- Two-phase parsing yields substantial thread-scaling boost; close to perfect up to 120 threads, with mutex having smaller impact
Thread scaling: Bowtie

- As with Bowtie 2, near-ideal scaling with queuing and cohort locks
Thread scaling: HISAT unpaired

- Huge improvements with queuing_lock and two-phase parsing
Thread scaling

- Further gains possible with batch parsing, where the first phase “lightly” parses several reads at once, reducing # critical section entrances
Experiment conducted by John Oneill at Intel

TBB + optimized parsing yields speedups of 1.1x - 1.8x on 88 threads on Broadwell E5-2699 v4 part. TBB/JHU Cohort lock outperforms other mutexes.
Experiment conducted by John Oneill at Intel

TBB + optimized parsing yields speedups of 2x - 2.7x on 192 threads on KNL B0 bin3 part. TBB/JHU Cohort lock outperforms other mutexes.
Thread scaling: summary

- Using a queue mutex / cohort lock can yield big improvement over spin / normal lock.

- Achieved near-ideal scaling up to 120 threads with (a) queue/cohort locks and (b) cleaner parsing for Bowtie, Bowtie 2.

- Promising scaling results on KNC & KNL; more to do.

- Cohort locks were best option in Broadwell & KNL experiments.

- Cohort locks seem to put KNL in a better position to outperform Xeon on genomics workloads.
Vectorization of Bowtie 2 inner loop

- Dynamic programming alignment not unique to Bowtie 2
- Common to many sequence alignment problems
Vectorization of Bowtie 2 inner loop

Outer loop iterates over columns
Main loop fills
Fixup loop adjusts, taking intra-chunk dependencies into account

Vectorization of Bowtie 2 inner loop

The wider the vector word, the more times the fixup loop iterates

- Mitigates the benefit of having wider words
Vectorization of Bowtie 2 inner loop

...but in some situations, the fixup loop can be skipped with little or no downside.

- Important future work is to determine whether selective suppression of fixup loop can remove most or all of the downside of having wider words.
Impact on the field

- As of Bowtie 1.0.1 release / Bowtie 2 2.2.0 release, Intel improvements are “in the wild,” assisting life science researchers.
Impact on the field

- Added TBB to Bowtie 1.1.2, Bowtie 2 2.2.6. Also added to public branch of HISAT. Plan to make TBB the default threading library in upcoming release.
Impact on the field

- Daehwan Kim of JHU IPCC team parallelized the index building process in Bowtie 2; TBB version of parallel index building available as of 2.2.7.
Impact on the field

- With changes fully reflected in Bowtie 1.2.0 and Bowtie 2 2.3.0, JHU team drafting manuscript describing improvements and lessons learned.
Future directions

- Where and why does the cohort lock help?
- Does cohort lock have a future in TBB?
- Can selective suppression of Bowtie 2 fixup loop unlock power of wider vector words?
- Can all of the above yield a big Knight’s Landing throughput win?
Other resources

- http://www.langmead-lab.org
- https://www.coursera.org/learn/dna-sequencing
Thank you

- John Oneill, Ram Ramanujam, Kevin O’leary, and many other great Intel engineers we spoke to and worked with
- Lisa Smith, Brian Napier and others in IPCC program
- Langmead lab team: Chris Wilks, Valentin Antonescu
- Salzberg lab team: Steven Salzberg, Daehwan Kim
- Intel
Thank you for your time

Ben Langmead
langmea@cs.jhu.edu
www.intel.com/hpcdevcon