C++ developer opening at Prime Seven

Gene Panov

What is an “electronic market-maker”

Then: floor market maker (specialist)

Now: Electronic market maker

How does an electronic market-maker work?

Huge amounts of
Price, Trade and other Data
Arrives (UDP, sometimes TCP)

Our commands
(quotes and orders)
go to exchange machines

NIC 1

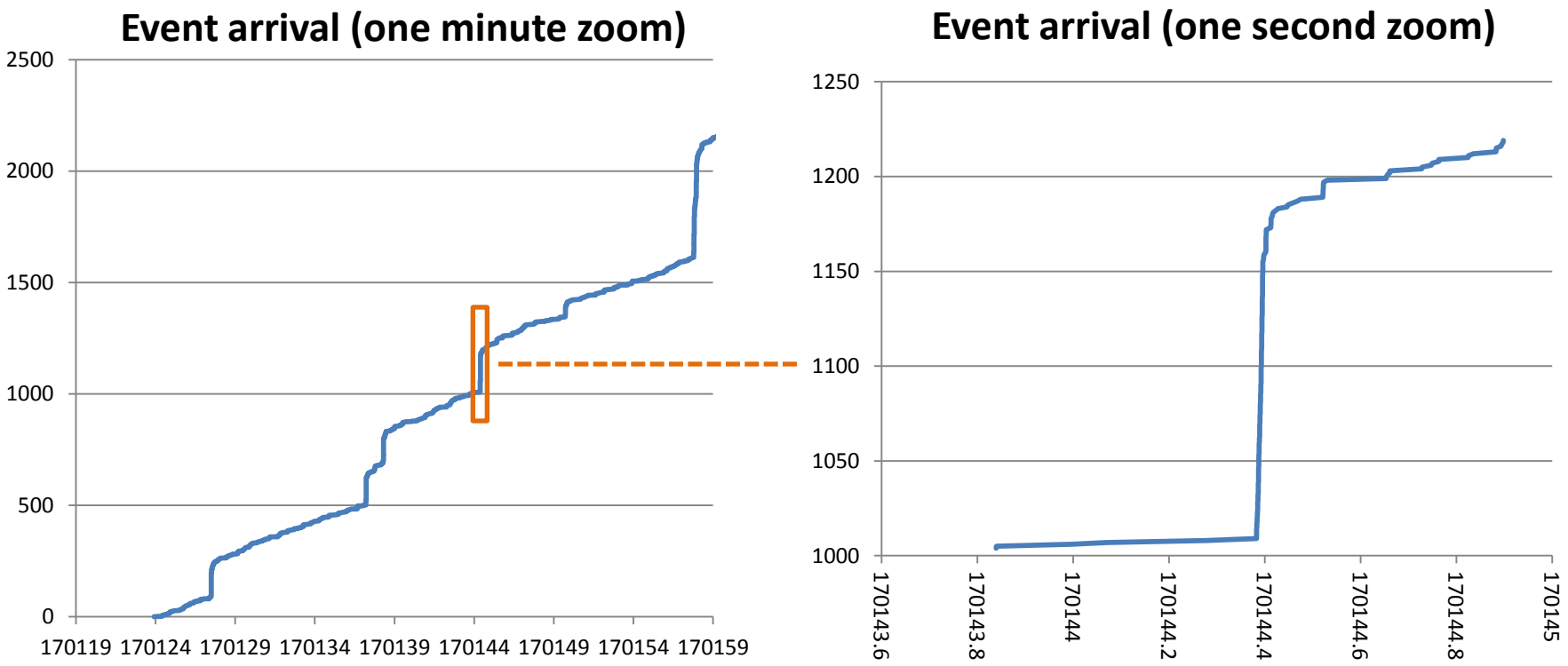
NIC 2

NIC 3

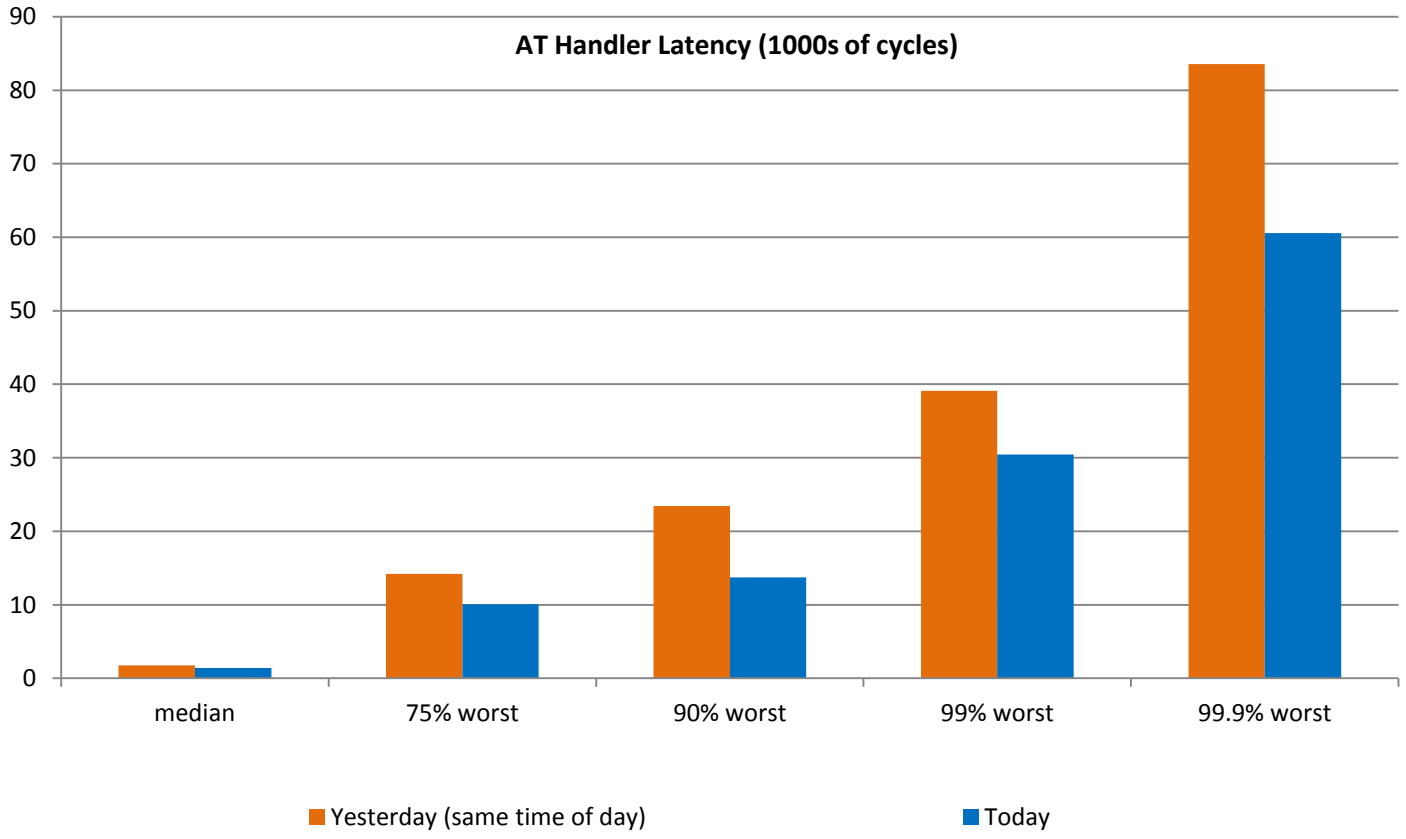
(typical machine)

Updates from slow systems
and operators arrive

Arriving data is very “bursty”

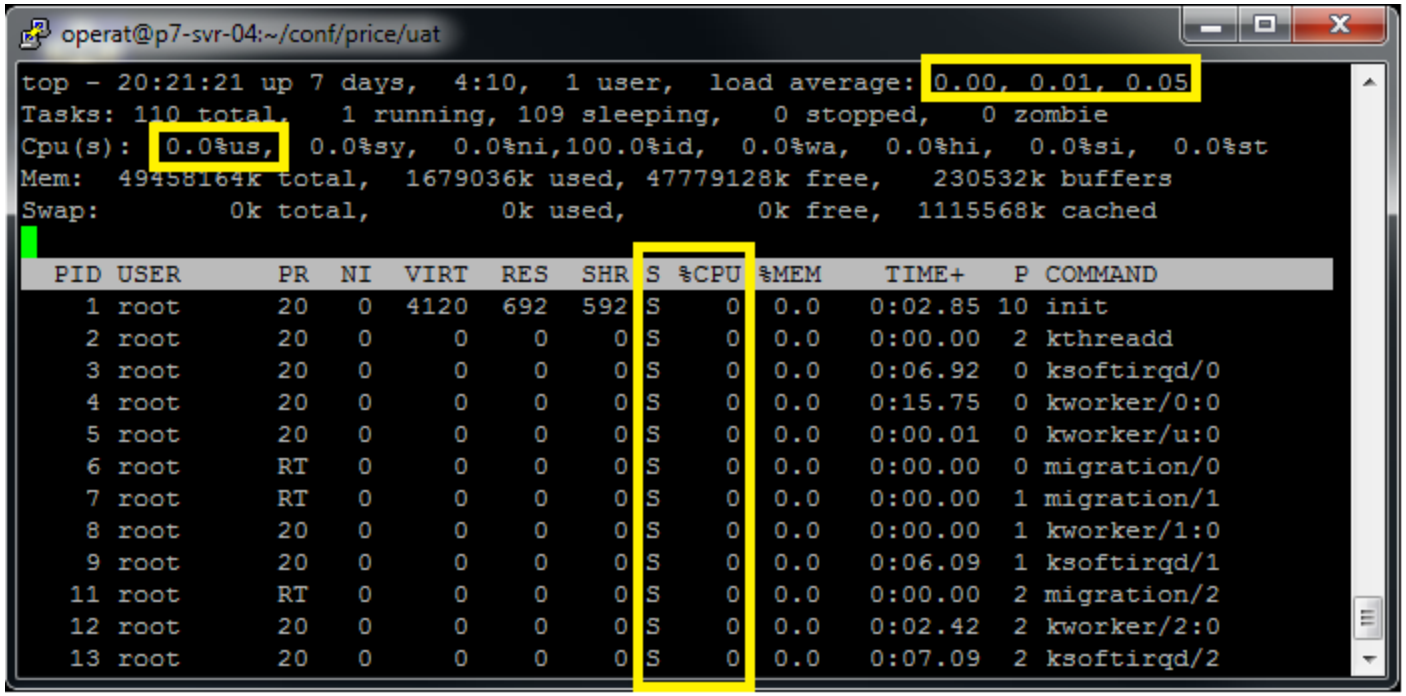


Arriving data is very “chunky”



Most of the time machines are idle (those not doing busy wait)

Dude, where's my process???



(not in the top lines of the "top" output, because the incoming data is "bursty", so most of the time CPU utilization is ~zero, but during a burst it's 100% for cores used)

Quiz: which queue gives lower latency?

```
 struct queue1 { // not thread-safe if >1 producer or >1 consumer
    void insert(int64_t value) {
        while (!empty_) { /* spin */ }
        value_ = value;
        empty_ = false;
    }
    bool try_consume(int64_t& target) {
        if (empty_)
            return false;
        target = value_;
        empty_ = true;
        return true;
    }
    queue1() : empty_(true) {}
private:
    // cache line 1
    volatile int64_t value_;
    char padding_[64 - sizeof(int64_t)];
    // cache line 2
    volatile bool empty_;
}
__attribute__((aligned(64)));
```

Quiz: which queue gives lower latency?

```
struct queue2 { // not thread-safe if >1 producer or >1 consumer
    void insert(int64_t value) {
        while (consume_counter_ != insert_counter_) { /* spin */ }
        value_ = value;
        ++insert_counter_;
    }
    bool try_consume(int64_t& target) {
        if (insert_counter_ == consume_counter_)
            return false;
        target = value_;
        consume_counter_ = insert_counter_;
        return true;
    }
    queue2() : insert_counter_(0), consume_counter_(0) {}
private:
    // cache line 1
    volatile int64_t value_;
    char padding1_[64 - sizeof(int64_t)];
    // cache line 2
    volatile uint64_t insert_counter_;
    char padding2_[64 - sizeof(uint64_t)];
    // cache line 3
    volatile uint64_t consume_counter_;
}
__attribute__((aligned(64)));
```

Let's find out…

```
/* worker thread */
template <typename queue>
struct worker {
    void operator() () {
        do {
            int64_t input_value;
            while (!input_.try_consume(input_value)) { /* spin until retrieved an input */ }
            int64_t result_value = input_value * input_value;
            output_.insert(result_value); /* insert the output */
        } while (true);
    }
    worker(queue& input, queue& output) : input_(input), output_(output) {}
private:
    queue& input_;
    queue& output_;
};

/* main thread */
template <typename queue>
void test_queue() {
    queue input, output;
    latency_curve histogram;
    worker<queue> worker_functor(input, output);
    boost::thread worker_thread(worker_functor);
    for (int64_t i = 0; i < 1000000; ++i) {
        thread_clock stopwatch;
        input.insert(i); /* insert an input */
        int64_t o;
        while (!output.try_consume(o)) { /* spin until retrieved an output */ }
        uint64_t cycles = stopwatch.cycles_since_start();
        histogram.add(cycles); /* how long did it take? */
    }
    std::cout << histogram.print() << std::endl;
    ::exit(0);
}
```

Results: it depends on hardware

Running on cores 1, 2 (different physical cores: same L3 cache, different L1,L2):

```
> sudo chrt -r 5 numactl --physcpubind=1,2 ./meetup_experiment 1
    (1000000pt) quantiles: 2858 2865 2888 2944 9501 15606
> sudo chrt -r 5 numactl --physcpubind=1,2 ./meetup_experiment 2
    (1000000pt) quantiles:  893  952 2969 3006 4974 12853
> #queue2 wins!
```

Running on cores 0 and 1 (hyperthreads of the same physical core, same L1,L2,L3):

```
> sudo chrt -r 5 numactl --physcpubind=0,1 ./meetup_experiment 1
    (1000000pt) quantiles: 375 375 375 375 390 21354
> sudo chrt -r 5 numactl --physcpubind=0,1 ./meetup_experiment 2
    (1000000pt) quantiles: 375 375 382 383 390 21382
> #queue1 wins! (but only by several cycles and only sometimes)
```

Running on cores 2 and 3 (hyperthreads of the same physical core, same L1,L2,L3):

```
> sudo chrt -r 5 numactl --physcpubind=2,3 ./meetup_experiment 1
    (1000000pt) quantiles: 375 375 375 375 380 19725
> sudo chrt -r 5 numactl --physcpubind=2,3 ./meetup_experiment 2
    (1000000pt) quantiles: 375 375 382 383 383 21163
> #queue1 wins! (but only by several cycles and only sometimes)
```

Prime Seven needs a C++ developer

Ideal candidate:

- Knows C++ well
- Knows sockets and threads
- Has “mechanical sympathy”

Evgeny.Panov@Prime-Seven.Com, 312-638-5177

Gene.Panov@Gmail.Com, 404-717-3266