Topic 7
Electronic Market Making
AIM WEEK 2016

Team:
Onur Çatmabacak, Mischa Knabenhaus,
Lichen Liang, Jianfang Lin,
Niccolò Moretti, Tiziano Müller,
Davide Torlo, Maria Veiga

Supported by:
Dr. Marelli @ ETHZ
Owain Lloyd, Hugo Hudson @ JPMorgan
“Time = Money”

– D. T.
“\[ \frac{1}{\text{Time}} = \text{Money} \]”

– Onur
Introduction

Problem:

Order book keeping for High Frequency Trading
Introduction

Problem:

**Order book** keeping for High Frequency Trading

Keeps track of buyers and sellers for a set of financial instruments
Introduction

Problem:

Order book keeping for **High Frequency Trading**

- high speeds
- high turnover rates
- high order-to-trade ratios
- high-frequency financial data

dealing with ~500ns for complete time-to-answer
Introduction
Introduction
Introduction

Storing & evaluating trades
Outline

• Simulated data
• Profiling
• Optimisation
  • Data structure
  • Caching
• Results
• Conclusion & Further developments
Simulated Data

**Stocks** $\sim O(1000)$

**Price levels** $\sim 100\sim4000$

Exchange **Quantities** $\sim O(10)$
Simulated Data

**Stocks** $\sim O(1000)$

**Price levels** $\sim 100\sim4000$

**Exchange Quantities** $\sim O(10)$

Use these insights to constrain the size of data structures
Profiling

AIM_code
Total samples: 2186
Focusing on: 2186
Dropped nodes with <= 10 abs(samples)
Dropped edges with <= 2 samples

__libc_start_main
0 (0.0%) of 2162 (98.9%)
main
62 (2.8%) of 2162 (98.9%)
__start
0 (0.0%) of 2162 (98.9%)
2162
std ostream
flush
1 (0.0%) of 1040 (47.6%)
generate_test_data
199 (9.1%) of 773 (35.4%)
std _Rb_tree
_M_insert_unique
159 (7.3%) of 186 (8.5%)
std ostream
_M_insert
34 (1.6%) of 167 (7.6%)
std ostream put
8 (0.4%) of 12 (0.5%)
std _Rb_tree_rebalance_for_erase
11 (0.5%) of 18 (0.8%)
__write_nocancel
1014 (46.4%)
gperftools valgrind
Optimisation

\[ O(\log N) \]

Stocks

Price level

sorted
Optimisation

<table>
<thead>
<tr>
<th>Stock</th>
<th>$100</th>
<th>$200</th>
<th>$300</th>
</tr>
</thead>
<tbody>
<tr>
<td>google</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>apple</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>trump</td>
<td>1</td>
<td>1</td>
<td>666</td>
</tr>
<tr>
<td>‘murica</td>
<td>0</td>
<td>999</td>
<td>999</td>
</tr>
</tbody>
</table>
Optimisation

Minimise the address space (reduce memory)

Price- vs Stock-Major order

A: Column-major order (Fortran-style)  B: Row-major order (C-style)
Optimisation

Minimise the address space (reduce memory)

Price- vs Stock-Major order

A: Column-major order (Fortran-style)  B: Row-major order (C-style)
Optimisation

stocks

price levels
Optimisation

stocks

price levels
Results
Results
## Results

<table>
<thead>
<tr>
<th></th>
<th>Price Major</th>
<th>Stock Major</th>
<th>Original</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>shift</td>
<td>no shift</td>
<td>shift</td>
</tr>
<tr>
<td>char</td>
<td>~190ms</td>
<td>~53ms</td>
<td>~430ms</td>
</tr>
<tr>
<td>int16</td>
<td>~210ms</td>
<td>~51ms</td>
<td>~590ms</td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th></th>
<th>Price Major</th>
<th>Stock Major</th>
<th>Original</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>shift</td>
<td>no shift</td>
<td>shift</td>
</tr>
<tr>
<td>char</td>
<td>~190ms</td>
<td>~53ms</td>
<td>~430ms</td>
</tr>
<tr>
<td>int16</td>
<td>~210ms</td>
<td>~51ms</td>
<td>~590ms</td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th></th>
<th>Price Major</th>
<th>Stock Major</th>
<th>Original</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>shift</td>
<td>no shift</td>
<td>shift</td>
</tr>
<tr>
<td>char</td>
<td>~190ms</td>
<td>~53ms</td>
<td>~430ms</td>
</tr>
<tr>
<td>int16</td>
<td>~210ms</td>
<td>~51ms</td>
<td>~590ms</td>
</tr>
</tbody>
</table>

Min runtime per iteration: $1.9 \times 10^{-5}$ ms

Max runtime per iteration: $3.8 \times 10^{-1}$ ms

Mean runtime per iteration: $5.1 \times 10^{-6}$ ms
## Results

<table>
<thead>
<tr>
<th></th>
<th>Price Major</th>
<th>Stock Major</th>
<th>Original</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>shift</td>
<td>no shift</td>
<td>shift</td>
</tr>
<tr>
<td>char</td>
<td>~190ms</td>
<td>~53ms</td>
<td>~430ms</td>
</tr>
<tr>
<td>int16</td>
<td>~210ms</td>
<td>~51ms</td>
<td>~590ms</td>
</tr>
</tbody>
</table>

- **Min runtime per iteration:** $1.9 \times 10^{-5}$ ms
- **Max runtime per iteration:** $3.8 \times 10^{-1}$ ms
- **Mean runtime per iteration:** $5.1 \times 10^{-6}$ ms
Conclusion & Further Developments

• Dealing with ‘warm up’ phase
Conclusion & Further Developments

- Dealing with ‘warm up’ phase
Conclusion & Further Developments

• Dealing with ‘warm up’ phase

• Data organisation
  • Alignment & Compression
  • Exploit stripe structure

• Maintenance work after generating metric

• Better time measuring tools (PAPI, HPET)
THE END

Trader

what my friends think
what my mom think i do
what society think i do
what market makers
think i do
what i think i do
what i really do