Brahma.FSharp + MailboxProcessor performance test report

Results of performance test of GPGPU calculation using Brahma.FSharp and MailboxProcessor composition are presented. Problem to solve is substring matching for data carving. Rabin-Karp algorithm was implemented using Brahma.FSharp for substring matching. F# MailboxProcessor used for composing of data reading, data processing on GPGPU, and data processing on CPU. Library for fast and flexible configuration of MailboxProcessors was created. Set of templates for search was fixed. Tests were performed for HDD and SSD storages. Low level sequential reading was implemented. First 16.5 Mb was processed.

Implementation available on GitHub: https://github.com/gsvgit/Brahma.FSharp/tree/Agents

Steps to run:

1. Clone https://github.com/gsvgit/Brahma.FSharp.git
2. Checkout Agents branch
3. Open \Brahma.FSharp\Source\Brahma.sln
4. Build it
5. Set Brahman.Runner as startup project
6. Note: we use system API to read disk, so application should be run with admin privileges.

Short sources description.

- Brahman.Runner — templates specification and entry point
- Brahman.Substrings — substrings matcher
  - Matcher.fs — entry point for Rabin-Karp on GPGPU
  - HDReader.fs — F# wrapper for disk system API
- Brahma.Agents — small library for infrastructure based on MailboxProcessors creation.

Configuration

- OS: Microsoft Windows 8.1 Pro
- System Type: x64-based PC
- Processor: Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz, 3601 Mhz, 4 Core(s), 8 Logical Processor(s)
- RAM: 16.0 GB
- HDD for test:
  - Model: ST3250410AS
  - Size: 232.88 GB
  - 7200 rpm
  -
- SSD for test
  - Model: INTEL SSDSC2BW240A4
  - Size: 223.57 GB
  - Max read speed: 540 Mb/sec

- GPGPU: NVIDIA GeForce GTX 560 Ti
    - CUDA Cores: 384
    - Core clock: 822 MHz
    - Shader clock: 1645 MHz
    - Memory data rate: 4008 MHz
    - Memory interface: 256-bit
    - Memory bandwidth: 128.26 GB/s
    - Total available graphics memory: 4095 MB
    - Dedicated video memory: 2048 MB GDDR5
    - Shared system memory: 2047 MB

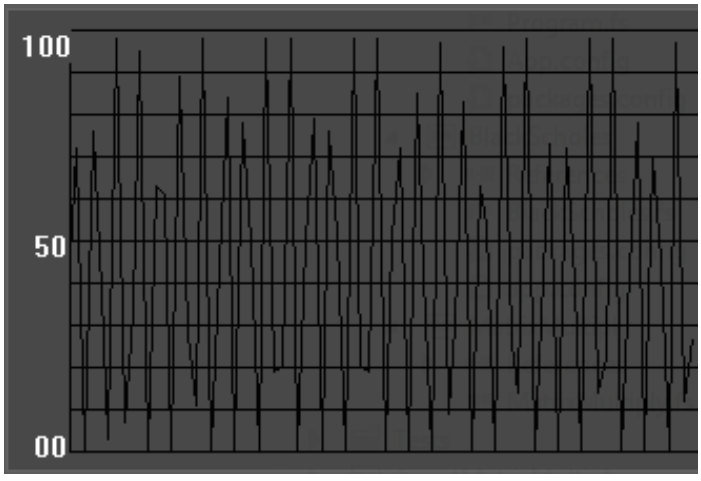
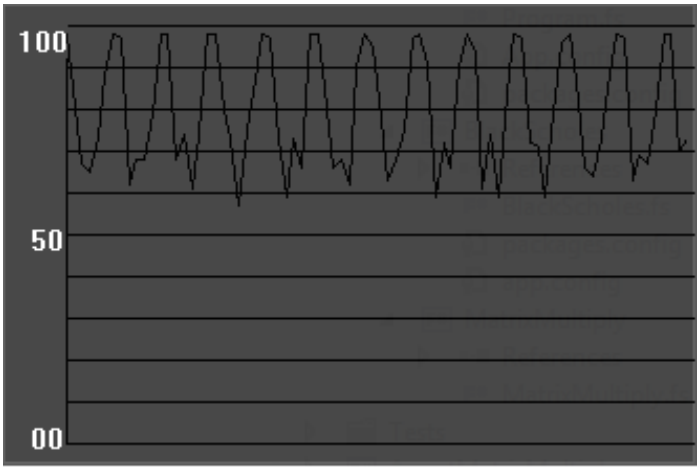
Results

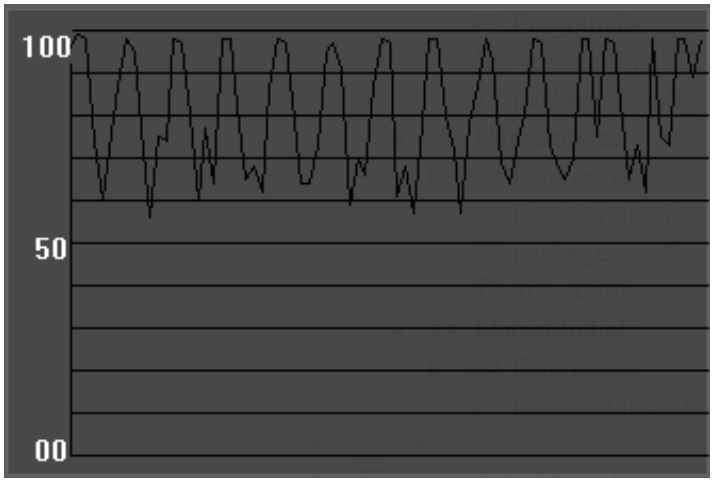
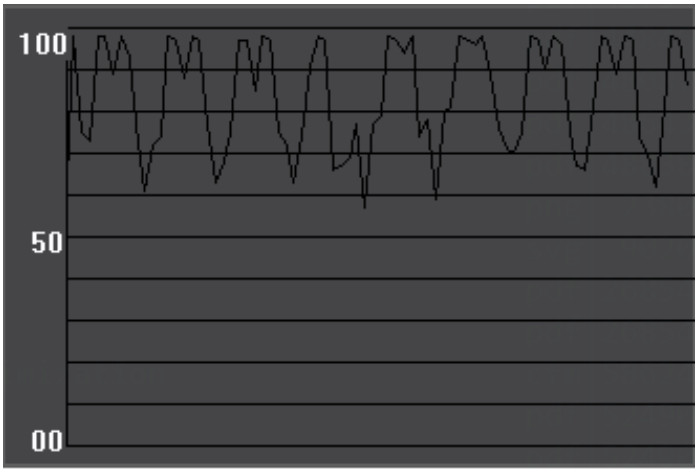
Tables below present results of tests.

- “buffers for data” — a number of arrays to fill by disc reader for each MailboxProcessor which communicate with GPGPU.
- “threads” — a number of MailboxProcessors which communicate with GPGPU. In current configuration we have only one GPGU, so all MailboxProcessors use it. For multi-GPGPU systems we can configure **k** MailboxProcessors for each GPGPU.

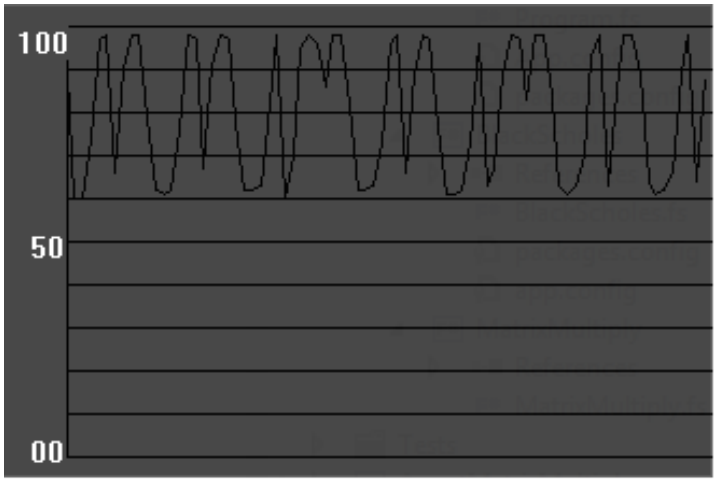
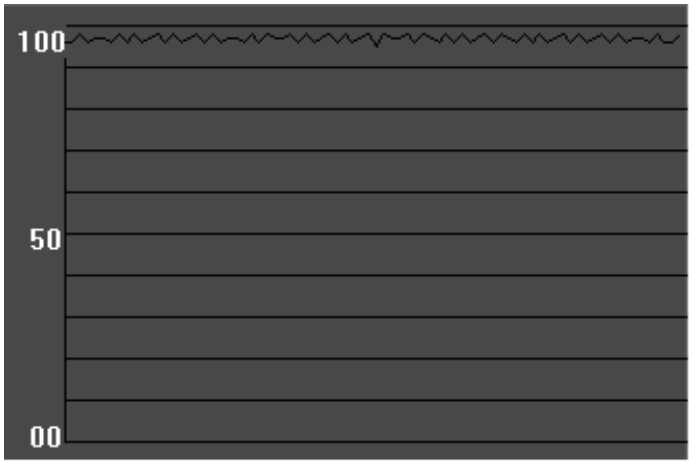
In each cell — total time and GPGPU loading graph.

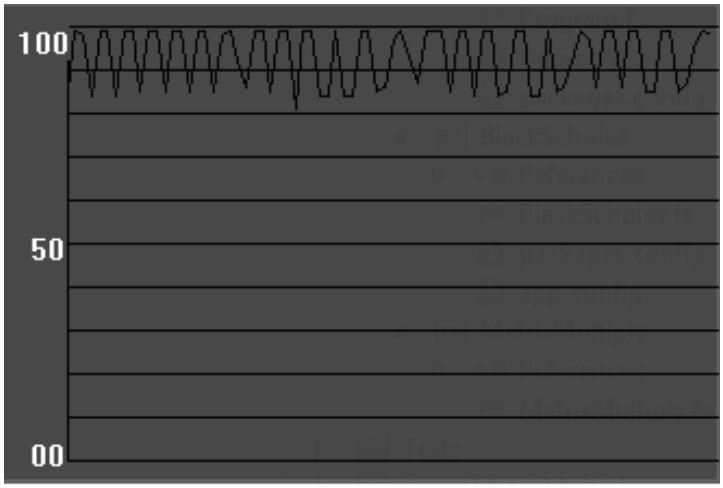
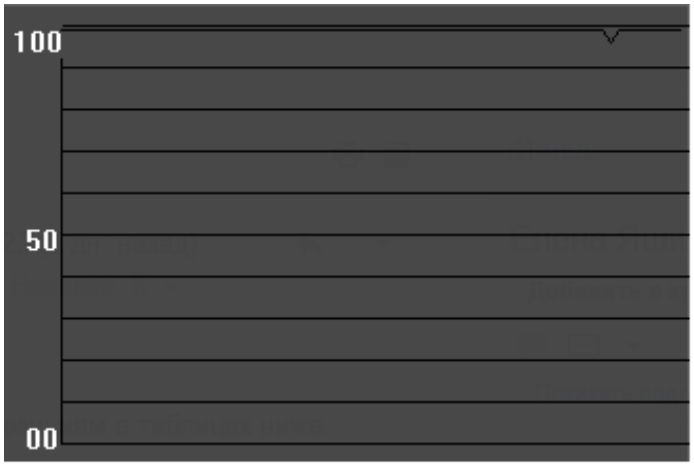
1. **HDD**

	1 buffer for data	2 buffers for data
1 thread	 Time: 316.0 sec	 Time: 173.9 sec

2 threads	 Time: 174.4 sec	 Time: 174.0 sec

2. **SSD**

	1 buffer for data	2 buffers for data
1 thread	 Time: 174.7 sec	 Time: 140.3 sec

2 threads	 Time: 145.7 sec	 Time: 138.0 sec

Conclusion

1. Data reading bufferization can sufficiently increase performance. Especially for HDD, where speed of reading is low.
2. For SSD processing with multi-GPGPU systems may be useful. Data reading is not so critical as for HDD and more than one GPGPU can be fully loaded by using flexible MailboxProcessors configuration. Configuration with two MailboxProcessors and two buffers for each of them can fully load one GPGPU.