AccelerIO

Accelerio | www.accelerio.com

Performance acceleration At Virtually no cost

Achindra Bhatnagar

2015

AccelerIO

In-Memory Computing Platform. It is built with in-memory storage and parallel execution engine. The design builds a highly resilient system.

A disk carved out of RAM is as fast as the RAM itself. With lowering costs of physical RAM and new systems coming up with higher and faster RAM modules, we intent to reserve and expose a portion of it as Virtual Disk Drives where we can host business critical data that can be fetched and processed several times faster than a hard drive or even an SSD.

Given the volatility of RAM, at this point we plan to deal with read-only data and thus not looking at making data backed by HDD/Swap partitions. However, we may want to back the RAM with redundant power supply so it can act as Non-Volatile RAM for a time we can guarantee its content to be backed up.

The initial system design for the PoC requires a kernel mode driver to reserve a large portion of RAM and expose it as a VHD. On this VHD we will run an instance of operating system (Windows Server Core). Another VHD carved in similar fashion will host data.

A Job manager is to be designed from the Server core running on one of the VHDs to provide job infra for users to submit their custom job to run against the data hosted by another VHD.

These VHDs are to be used as pass-through disks so we can minimize any OS component overhead.

Later version will bring

1. Support for read-write data,
2. Host this software in Cloud,
3. Job redundancy across geo/clouds so we can guarantee job completion if one geo goes down
4. Expose AccelerIO as SaaS where Job infra can be subscribed and business critical data can be hosted in cloud
5. Appliance for customer in-house needs
6. Use of commodity hardware to cluster and run AccelerIO on it

Our solution will enable existing software to be accelerated without any customization, thus there are a wide variety of customers for this kind of system.

While we are able to speed up compute by a factor of x5000 by removing the disk fetch latencies completely and this software can be hosted in cloud and accessed as a cloud service, several real time applications can make use of this, like analytics, mobile computing, IoT, etc.

This is a new business and the team will start by investing their personal time to build the PoC.

# Executive Summary

At AccelerIO we are solving IO bottleneck which results in acceleration in compute. We have a software solution to reduce the latency in disk IO where data is stored and result in an upscale of computer by several factors.

There are various products in market which try to address this problem in various ways. Our product stands apart from our competitors in terms of a scalable and several factor improvements in performance at a relatively lower price of ownership.

## Objectives

A PoC with results from various benchmarking should be available by September 2015. Followed by an in-cloud test (Azure) of PoC by October 2015. We should be able to run an OS from the AccelerIO along with software requiring high speed IO. This may include Image Processing application or excel macro calculation for example.

## Milestones

[TBD]

1. **AccelerIO v0.1 – [Date]**
   1. Provisioning service boots base OS from RAM disk
   2. Provisioning service expose large disk (>4GB) through its driver loaded in base OS
   3. Provisioning service loads data files to new exposed large disk
   4. Provisioning service launches Job manager on Base OS
   5. Web and API interface of Job Manager starts job workflow
   6. Real time performance counters exposed through provisioning service
2. **Early Preview to real Users – [Date]**

## Code

<https://accelerio.visualstudio.com/>

## Investors

Microsoft Ventures could be one potential investor who invests in Technology start-ups with objective of early adoption of Microsoft technologies. They provide ~$10k Azure credits and introduce big VC investors apart from mentoring and office space.

We may look for introductions in Silicon Valley as investors in India appears to be more focused on Web and retail market start-ups.

## Market Analysis / Market Segmentation

[TBD]

1. Market Opportunity
2. Competition (are people already doing this)
3. Validation with real users
4. What others are doing for performance boost?
   1. Flipkart, Stock Broking, etc

## Competition

<https://hazelcast.com/products/>

<http://go.sap.com/solution/in-memory-platform.html>

<http://www.gridgain.com/products/in-memory-data-fabric/>

[TBD]

## Pricing

[TBD]

* Stand-alone Software for high end machines
* Cloud hosted
* SaaS
* Appliance

## Capital and Expense

[TBD]

## Break-Even / Exit Plan

[TBD]

## Status

1. **8/12/2015**:
   * We are able to boot Windows XP from a disk carved out of memory and exposed as a VHD to Hyper-V. The boot time is considerably faster than from a normal disk. It took us 6 sec. to boot Windows XP. Another VM booting from hard disk based VHD took around 27 seconds.
   * Next is to PXE boot physical machine with RAMDisk option in boot loader and run our driver to expose the rest of memory as a data disk. At present our driver is limited to exposing 4 GB memory as disk, we will extend this later to larger disks.
2. **8/26/2015**
   * Disk performance statistics

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **Packet** | **Counter** | **Fast** | **SSD** | **Normal** | **Over SSD** | **Over Normal** | | **512B** | IOps | 16320 | 12560 | 348 |  |  | |  | MBps | 8.36 | 6.44 | 0.18 | **1.3x** | **46x** | |  | Avg Response ms | 0.06 | 0.08 | 2.86 |  |  | |  |  |  |  |  |  |  | | **4KB** | IOps | 14537 | 10588 | 350 |  |  | |  | MBps | 59.71 | 44.00 | 1.44 | **1.4x** | **41.5x** | |  | Avg response ms | 0.067 | 0.94 | 2.85 |  |  | |  |  |  |  |  |  |  | | **4KB aligned** | IOps | 14555 | 3499 | 349 |  |  | |  | MBps | 59.66 | 14.33 | 1.43 | **4.2x** | **41.72x** | |  | Avg response ms | 0.067 | 0.28 | 2.86 |  |  | |  |  |  |  |  |  |  | | **16KB** | IOps | 6975 | 6542 | 328 |  |  | |  | MBps | 114.42 | 107.75 | 5.37 | **1.1x** | **21.3x** | |  | Avg response ms | 0.14 | 0.15 | 3.05 |  |  | |  |  |  |  |  |  |  | | **32KB** | IOps | 3626 | 5103 | 310 |  |  | |  | MBps | 119.63 | 164.81 | 10.16 | **0.7x** | **37.15x** | |  | Avg response ms | 0.27 | 0.20 | 3.22 |  |  | |  |  |  |  |  |  |  | | **64KB** | IOps | 1895 | 3190 | 280 |  |  | |  | MBps | 124.31 | 204.94 | 18.37 | **0.6x** | **6.76x** | |  | Avg response ms | 0.53 | 0.32 | 3.57 |  |  | |  |  |  |  |  |  |  | | **256KB** | IOps | 499 | 1296 | 88 |  |  | |  | MBps | 130.84 | 342.44 | 23.20 | **0.4x** | **5.63x** | |  | Avg response ms | 2.00 | 0.78 | 11.29 |  |  | |  |  |  |  |  |  |  | | **Default** | IOps | 15191 | 4773 | 349.56 |  |  | | *67% read, 2KB* | MBps | 31.07 | 9.10 | 0.72 | **3.4x** | **43.15x** | | *Random* | Avg Response ms | 0.06 | 0.25 | 2.86 |  |  | |  |  |  |  |  |  |  | | **All in one - Average** | IOps | 3004 | 965 | 275 |  |  | | *512b* | MBps | 114.06 | 36.41 | 10.44 | **3.1x** | **10.92** | |  | Avg Response ms | 0.33 | 1.05 | 3.63 |  |  | |

*Good performance for small IO, performance degraded for large IO.*

TODO: Increase buffer size to 1M and test

## TASKS

AccelerIO platform will have following components

* **Miniport Driver,** 
  + Default creates disk of 50% RAM
  + Generate Perf counters
  + Setup scripts
    - One click installs and configuration
    - Default format to 4KB blocks
    - Default threads 4xProcessor cores
  + Should start at boot in background, no user interaction required
  + Package should be remote deployable given access to machine
    - Scripts to set certificates in place, reboots and restart driver

* **Control Application** 
  + Change configuration and changes must persist in registry
  + Control other driver instances from one application remotely
  + Consume perf counters
  + Format disk with various block sizes
  + Associate a backing Disk/VHD/Network Disk
  + A web dashboard for Monitoring and Control

* **Filter Driver**
  + Worker infra (async) to sync data between backing disk and fast disk
  + Redirect IO during system boot and sync
  + Control shutdown path for data persistence
  + Expose Perf Counters
  + Scale for latency in syncing with backing disk

* **Use Cases**
  + Identify workloads and appropriate interface to expose/access Fast Disk
    - Application data use pattern
      * Visualizer
      * Optimal Defrag
    - Host DB
      * BD files on Fast Disk
    - Host Applications
      * Application and Data on Fast Disk
    - Host a Cache
    - Host an OS
* **Later**
  + Debup
  + Tiering
  + Resilience (Error free/versioning)
  + Distribution
  + RAID\*
  + File and Block access
  + Transparent service migration
    - Transparent replication
  + Backup/Recovery/DR

## Use Cases

## Problems to Solve

1. Loading data to RAMDisk
   * A bitmap representing what part of data is loaded in RAMDISK, each IO is checked against this map to determine if it can pass to RAMDISK or redirected to backing device.
   * While a thread in background can continue to copy data from backing device to the RAMDISK and update the bitmap.
   * Similarly, a lazy writer thread can be employed to persist dirty data out of RAMDISK to backing store when we allow for data to be updated in RAMDISK.
   * System will initially perform at the same rate as that of the backing store, however soon this will start running at AccelerIO
2. RAMDISK consistency
   * RAM may develop permanent grey areas which will act as bad sectors of disk.
   * When data is copied from backing store to disk, it will be read back and compared with data from backing device for consistency check. Bitmap will be updated accordingly and an alert will be generated.
   * A counter of number of bad bits will be maintained to decide when to replace the RAM module for better performance.
   * Read requests for bad bits will be redirected by the bitmap to the backing store.
     + Will have to see if this data can be moved to another region and metadata updated. This can then be persisted in the backing device. Similar to recovering data from bad sector on a disk.