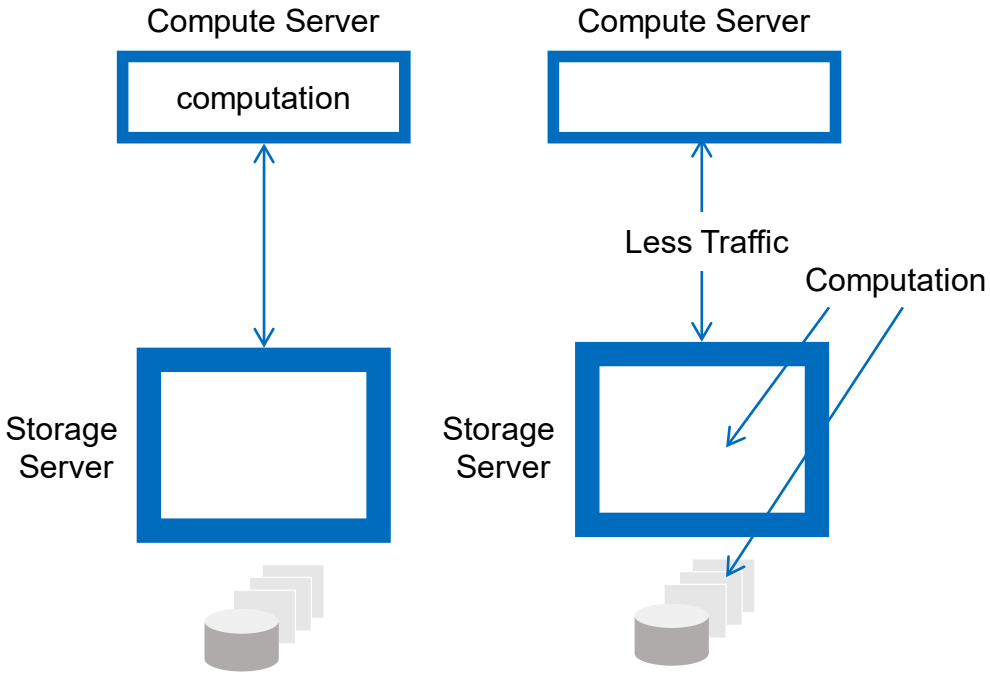
STORAGE DEVELOPER CONFERENCE
SDC 21
BY Developers FOR Developers
Virtual Conference
September 28-29, 2021
A SNIA Event

Computational Storage Directions at Fungible

Presented by Dr. Jai Menon, Chief Scientist, Fungible

Computational Storage - Background

- Move compute to the data instead of data to the compute
- Value
  - Less data transferred on the network
  - Faster response times
  - Improved security
  - Reduced physical footprint
- Architectural Approaches
  - Move compute into the drive
  - Move compute into the storage array
  - Compute platform on the NVMe bus or the NVMeoF network
- Implementation approaches
  - FPGA, GPUs, ASICs with embedded Arm, DPUs
- Standards
  - SNIA TWG and NVMe Computational Storage Task Group



Use Cases

Database Acceleration – Scans and aggregations close to data

Big Data Analytics – Generating insights directly on the data

Image Classification – Meta-tagging directly on the data

Smart Vehicles -- Direct processing of vehicle telemetry data

Science experiments – filtering close to the data

CDNs – data manipulation at the source for localization

Fungible's Approach Uses the Fungible DPU™

Fungible Offerings are Powered by the Fungible DPU

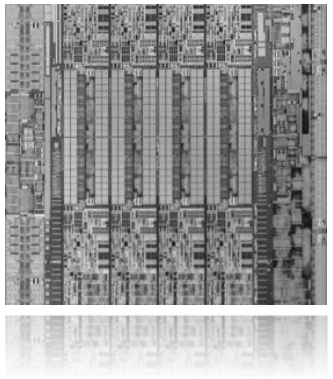
A New Class of Microprocessor Purpose-Built for the Data-Centric Era

The Fungible DPU is a new class of programmable microprocessor that:

Enables 10x more efficient execution of data-centric workloads

Implements a scalable, low tail latency, congestion-free TrueFabric™ endpoint

CPU

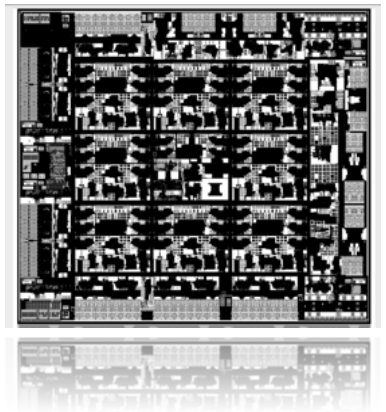


General-Purpose

GPU

Vector Floating Point

Fungible DPU™

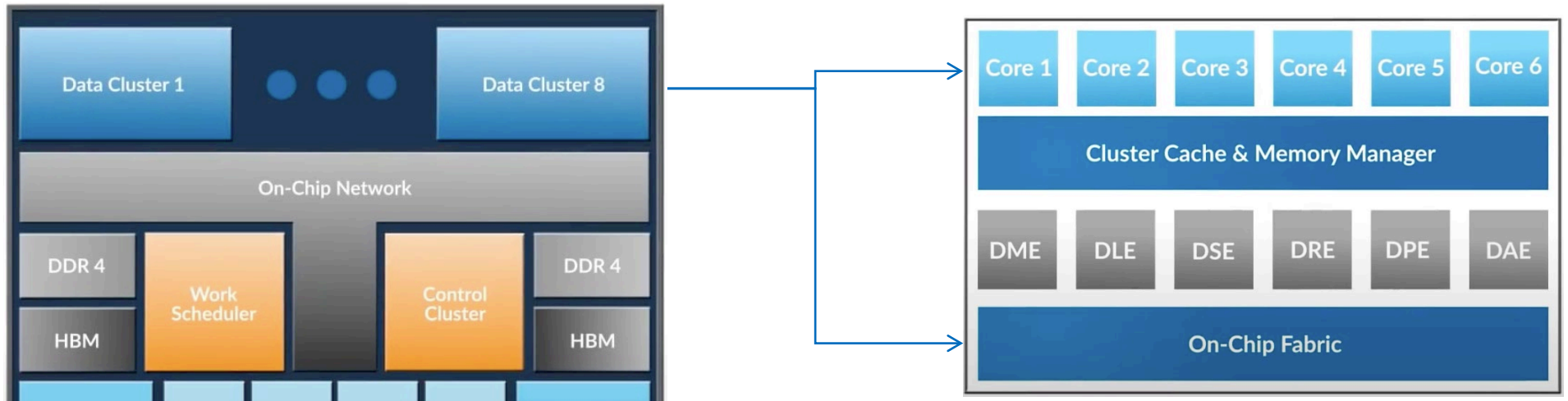


Data-Centric

- Standard external interfaces – PCIe, Ethernet
- Programmable in C
- Many multi-threaded cores & hardware accelerators for crypto, compression, etc.,

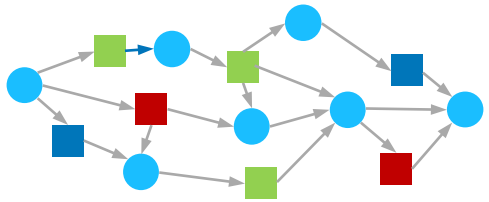
[1] Data-centric = stateful, multiplexed processing of high b/w data streams

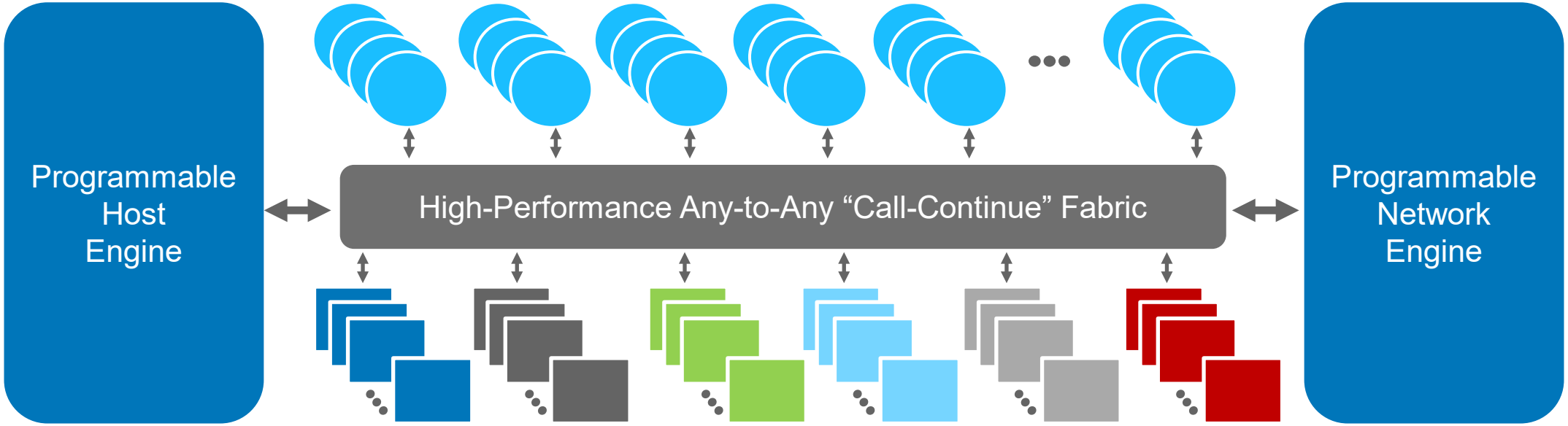
Fungible DPU has Lots of Cores and Lots of Accelerators



F1 DPU runs 192 hardware CPU threads and dozens of accelerators in parallel

Fungible's programming model makes computational storage easy





Traditional DPUs have loose coupling between cores and accelerated path

Computational Storage using DPU-based storage appliance

-

Cloud Data Center Requirements on Block Storage

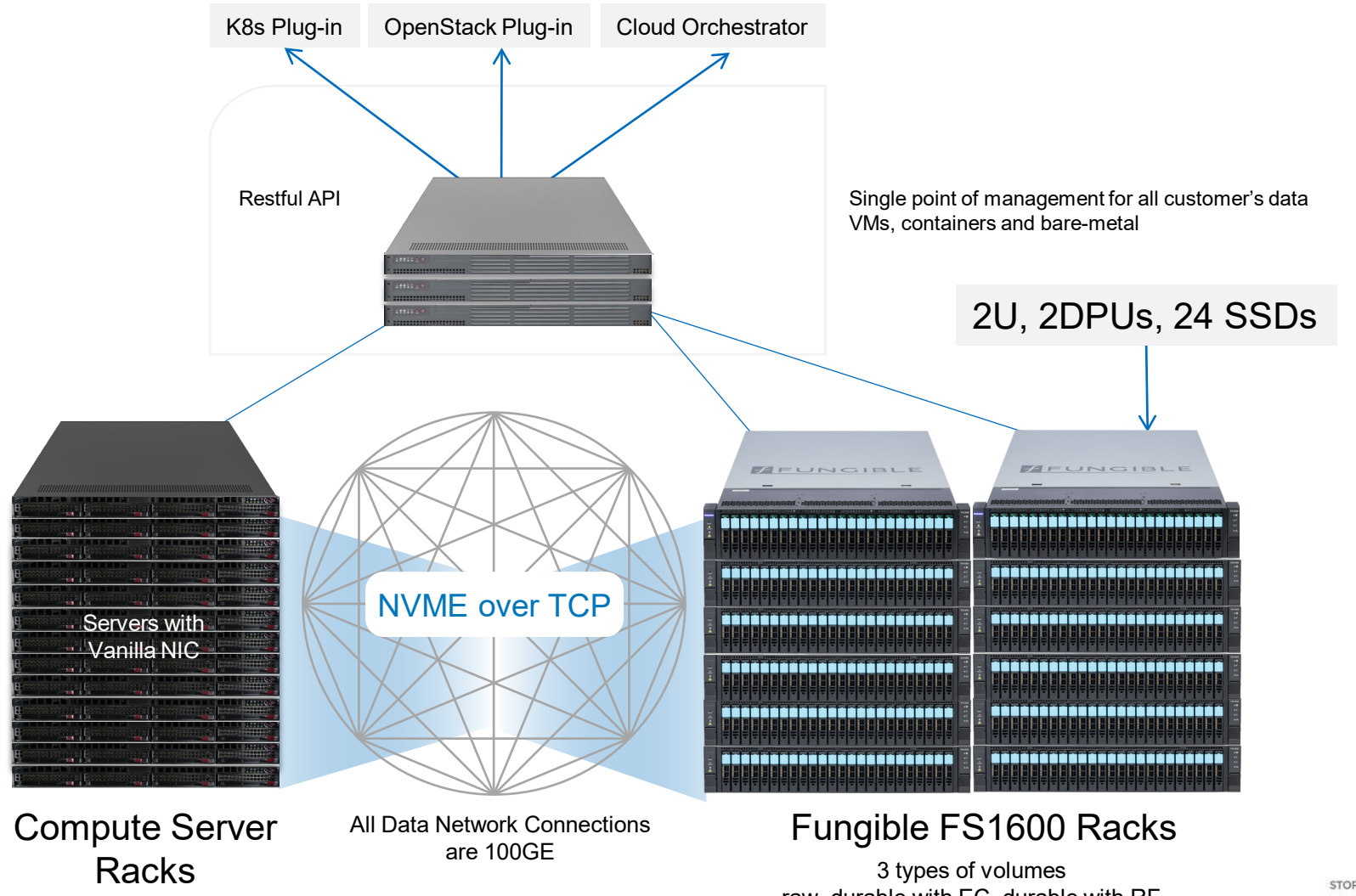
Requirement	Benefit
Storage is pooled and shared across all servers	High storage utilization; Independent storage scaling
Very high and consistent performance	Networked storage @ local SSD performance
Scale out	Grow as you need; Pay as you grow
Line-rate Compression	TCO; high storage utilization
Line-rate Encryption	Security; workload consolidation
Multi-tenancy (per vol protection, encryption, QoS)	Workload consolidation
REST API to manage PBs of data	TCO
Rack scale resiliency @ low overhead using networked EC	Very high reliability @ low cost
Supports VMs, containers, bare-metal	Workload consolidation

Blue rows need DPU

Fungible Storage using DPU

NVMEoF Based Elastic Block Store

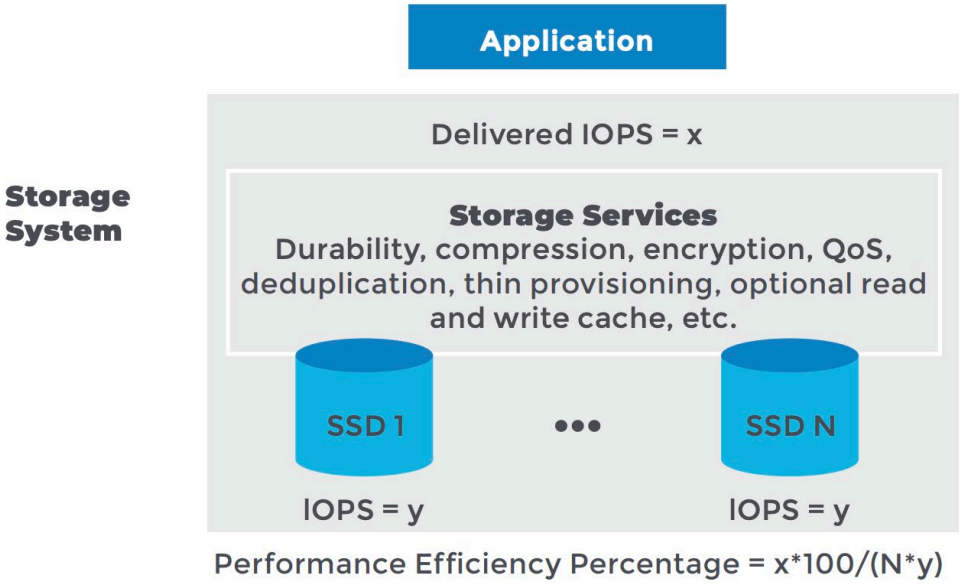
- HIGH PERFORMANCE
- LOW TCO
- SECURITY
- AGILITY
- SIMPLICITY



Performance Efficiency Percentage (PEP)

New Performance Metric for Storage Systems

Intuitive, easily measured, applies to all workloads



$$PEP = \frac{\text{Delivered IOPS}}{\text{\# of SSDs x SSD IOPS capability}}$$

Ideal PEP = 100%
FS1600 PEP with basic function = 95% (1 DPU to 12 SSDs)
FS1600 PEP with durability, compression, encryption, etc. = 70%

Expect high PEP with computational storage functions allowing for high-speed scans and filtering

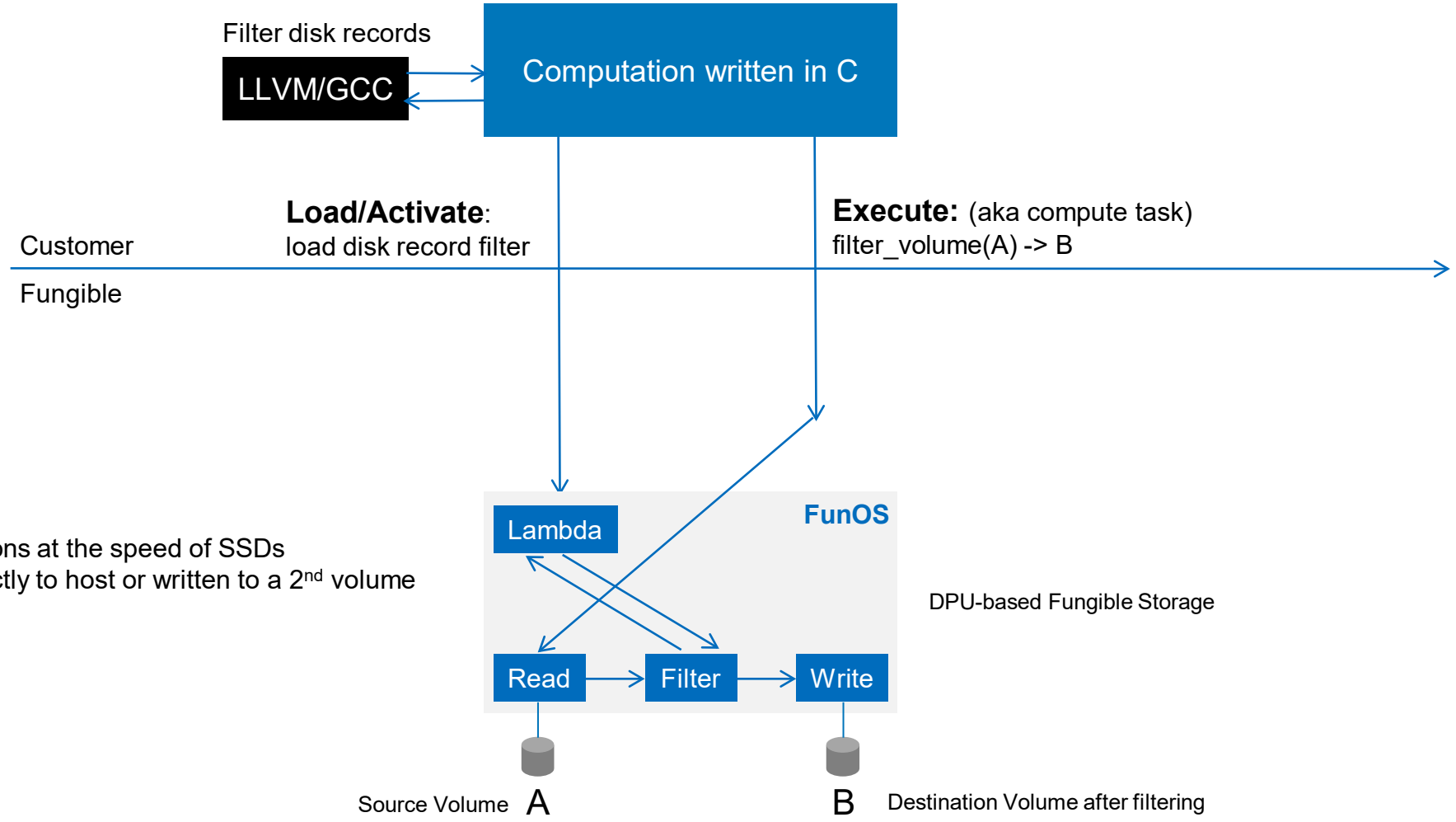
Computational Storage on FSC

3 Approaches to Computational Storage with the DPU

01	02	03
eBPF Style Approach	**Regex Pattern Matching**	**Domain specific language**
Computations written in C, downloaded to DPU, then called as needed to execute the downloaded code. Consistent with computational storage standard	Patterns to look for in a stream of data specified using PCRE (Perl Compatible Regular Expressions). These are compiled to state machine code which is downloaded and executed in the DPU	Computations expressed in a domain specific language such as SQL or Apache Spark. Compiled to code that runs in the DPU

Method 1 — eBPF style downloadable programs

eBPF Computational Storage



Execute scans and aggregations at the speed of SSDs
Filtered data can be sent directly to host or written to a 2nd volume

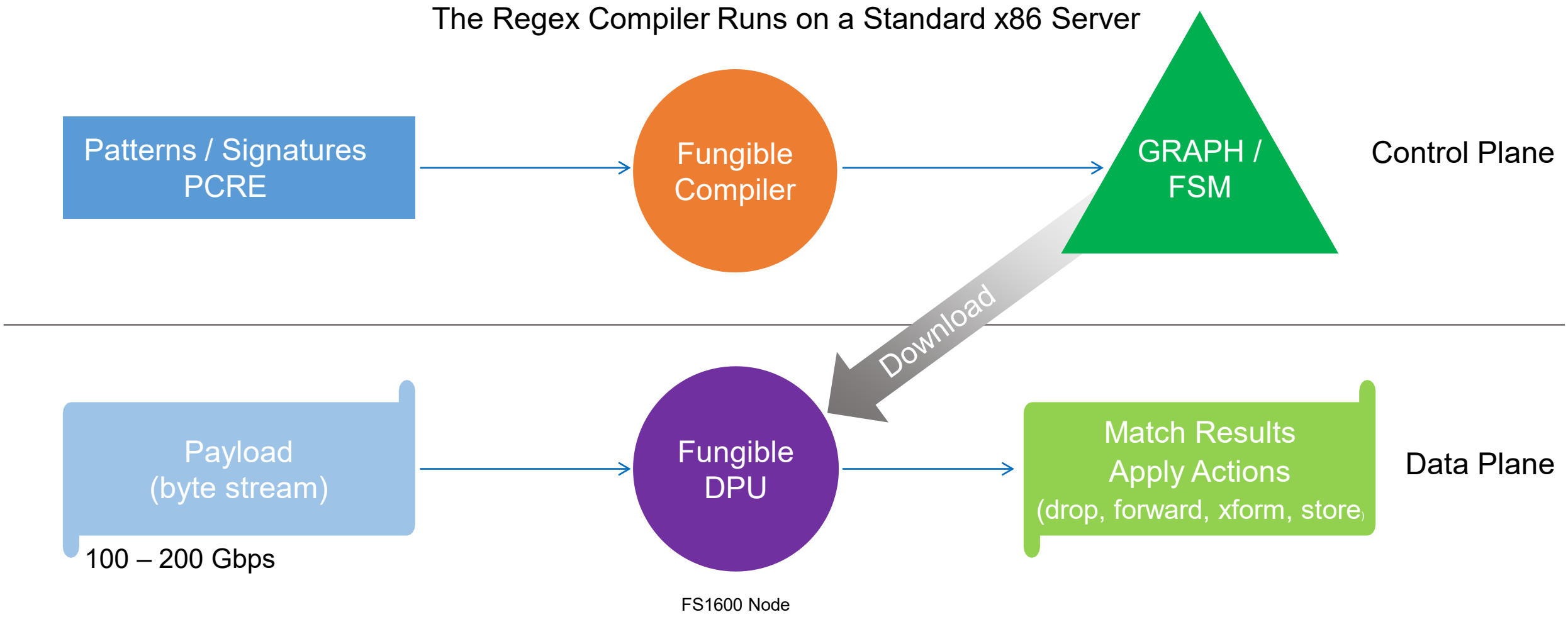
DPU-based Fungible Storage

Steps of the Workflow to Download and Execute Code

- Write a filter as C code. In our example, data from a source volume is filtered and written to a destination volume.
- Code is compiled using LLVM/GCC (creates executable & linkable format (elf)).
- Load elf filter code into FunOS running on the DPU
- Activate – insert new code into some existing dataflow
- Run the filter code
  - Source and target volumes are specified in the call
  - Length of each record to be examined is specified in the call
  - Code knows the format of fields within records
  - Source volume is read and destination volume is populated with filtered records

Method 2 — Regex Pattern Matching Computations Device-defined programs

Computation is Pattern Matching at High Speeds



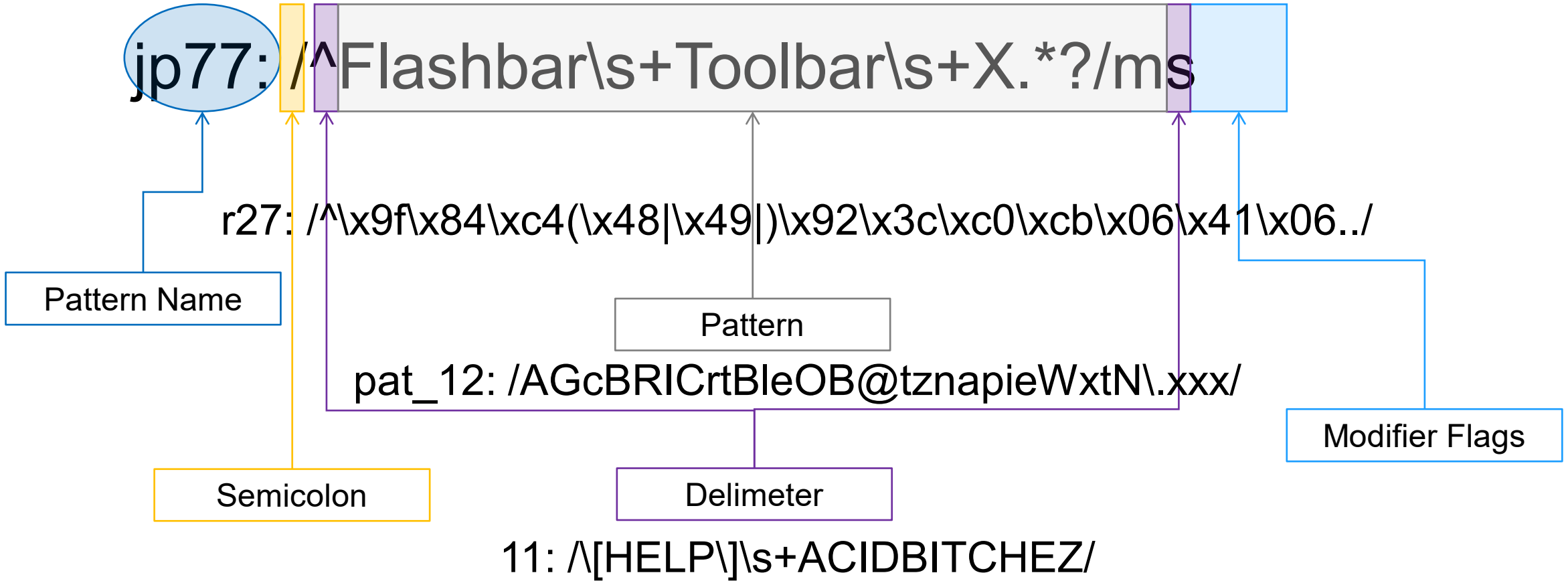
Supported PCRE Constructs

Supports most of the standard PCRE constructs. A few examples below.

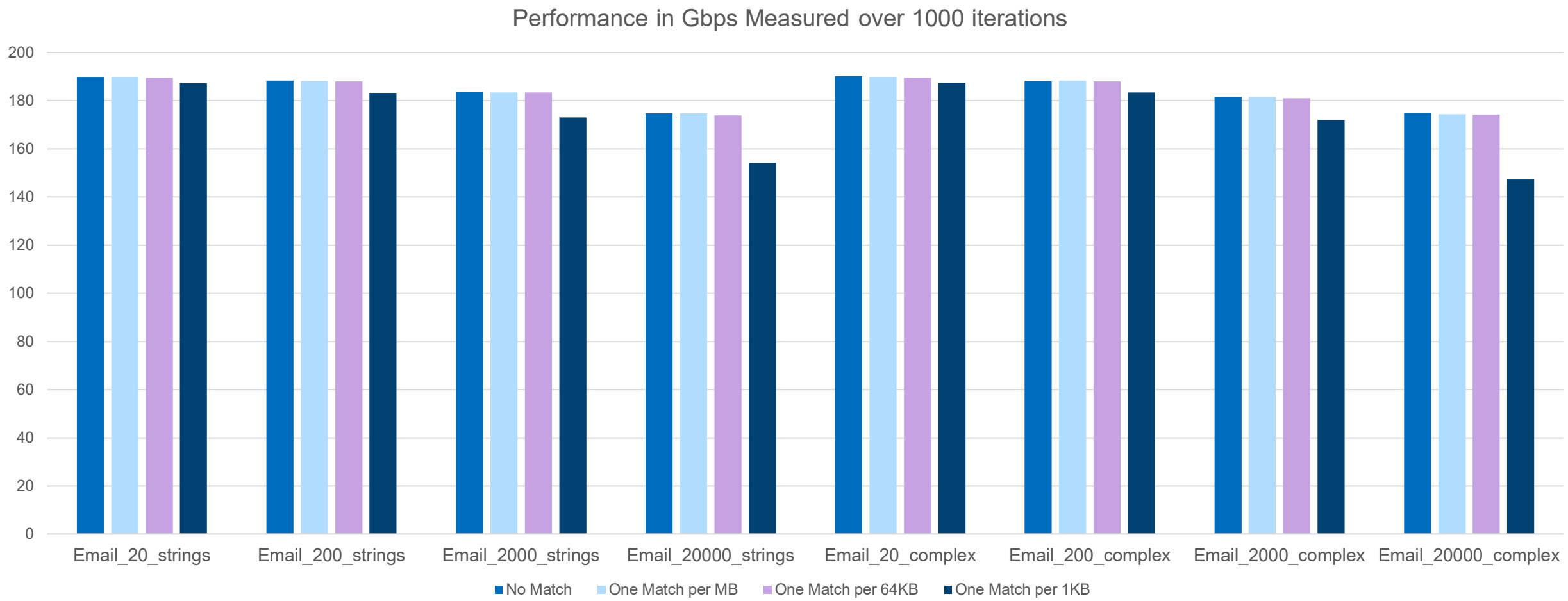
Characters:	Character Classes:	Anchors:
Case-sensitive, case-insensitive, non-printable (hex and octal representations) and special characters.	Sets, Ranges, Named classes, Short hands and negations of character classes.	Carats(^), dollars($), start of string (\A), end of string (\z, \Z), boundaries (\b, \B).

Alternations & Closures:	Grouping & Back references:	Assertions:
Alternations, closures (Both Lazy and Greedy), Optional, Repetitions.	Capturing and non-capturing grouping, Back references	Lookahead, lookbehind, anchored, offset, positive, negative.

Sample Pattern constructions

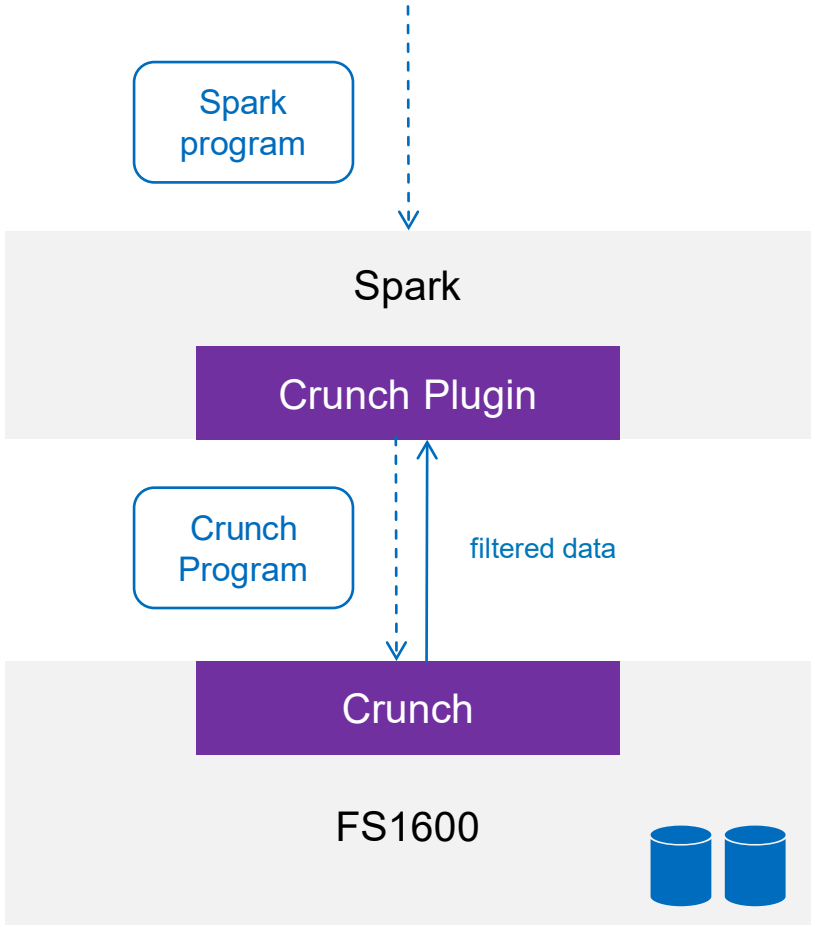


Performance Chart

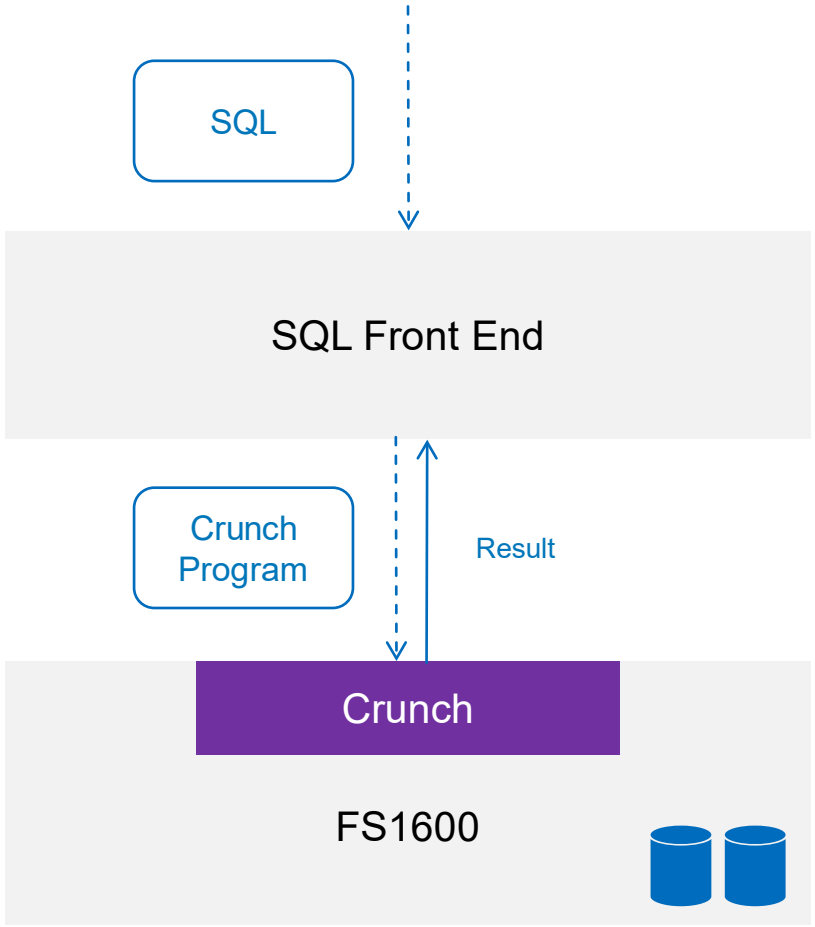


Method 3 — Express Computations in Domain Specific Languages

Executing Spark programs



Executing SQL Programs



Summary of Fungible's Approach to Computational Storage

- Computations are pushed to a storage appliance (FS1600) built using 2 DPUs and 24 SSDs
  - DPU has 192 threads tightly integrated with many multi-threaded accelerators
  - FS1600 has high PEP – computations at close to speed of many SSDs
- Computations can be expressed 3 ways
  - In C code (eBPF) – downloadable programs
  - As regular expression (PCRE) – device-defined programs
  - In a domain specific language like SQL
- Computational Storage is a natural extension of the Fungible DPU programming model

THANK YOU

Please take a moment to rate this session.

Your feedback is important to us.

