Scalable SA

Hal Rosenstock
May 19, 2015

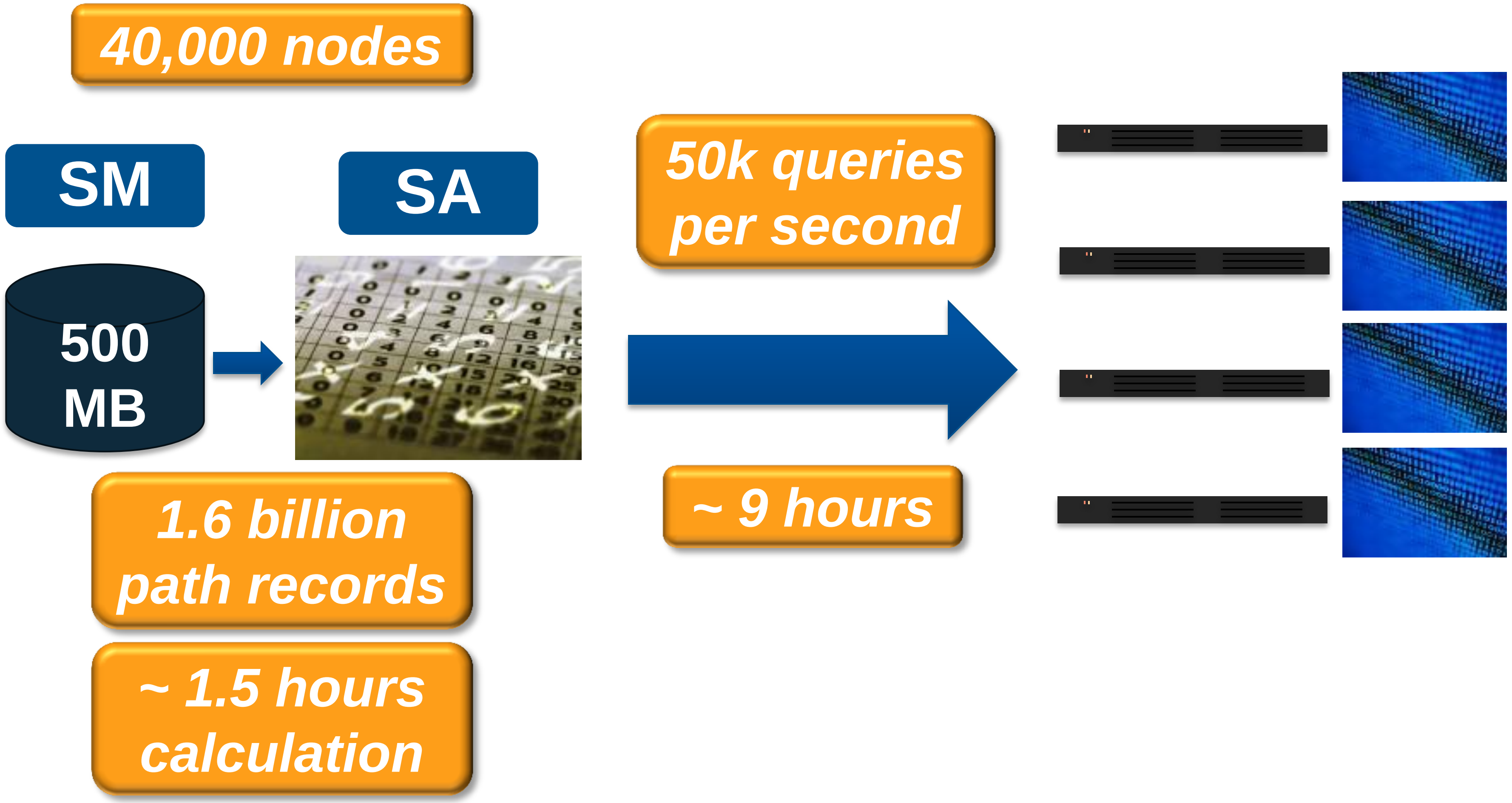
Mellanox® TECHNOLOGIES
Connect. Accelerate. Outperform.™

The Problem And The Solution

n^2 SA load

- SA queried for every connection
- Communication between all nodes creates an n^2 load on the SA
  - In InfiniBand architecture (IBA), SA is a centralized entity
- Other n^2 scalability issues
  - Name to address (DNS)
    - Mainly solved by a hosts file
  - IP address translation
    - Relies on ARPs
- Solution: Scalable SA (SSA)
  - Turns a centralized problem into a distributed one

Analysis



40,000 nodes

SM

SA

500 MB

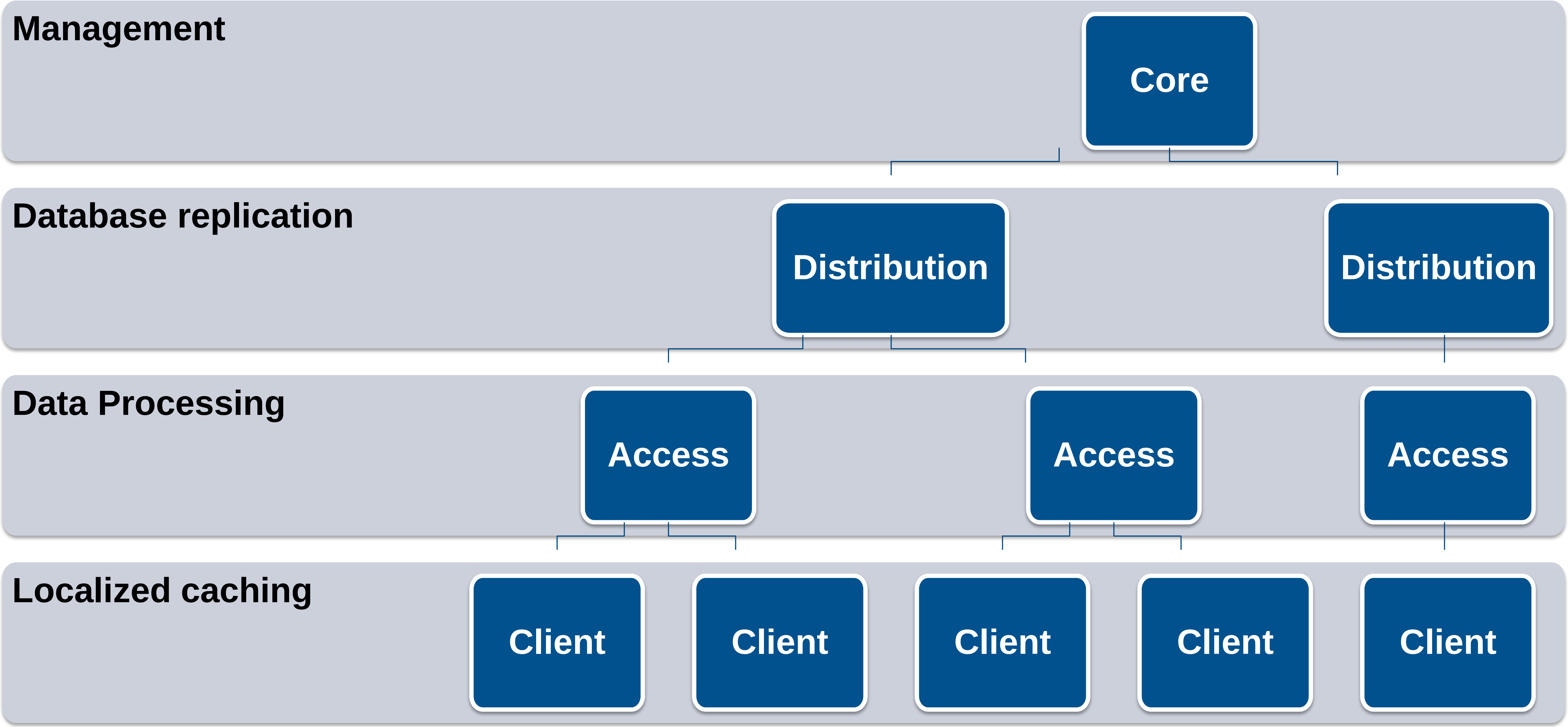
50k queries per second

1.6 billion path records

~ 1.5 hours calculation

~ 9 hours

SSA Architecture



Distribution Tree

- Number of management nodes needed is dependent on subnet size and node capability (CPU speed, memory)
  - Combined nodes
- Fanouts in distribution tree for 40K compute nodes
  - 10 distribution per core
  - 20 access per distribution
  - 200 consumer per access
- Built with rsockets AF_IB support
- Parent selected based on “nearness” based on hops as well as balancing based on fanouts

rsockets AF_IB rsend/rrecv performance

- On “luna” class machines as sender and receiver with 4x QDR links and 1 intervening switch
  - 8 core Intel(R) Xeon(R) CPU E5405 @ 2.00GHz
- Default rsocket tuning parameters
- No CPU utilization measurements yet
- SMDB: ~0.5 GB (for 40K nodes)

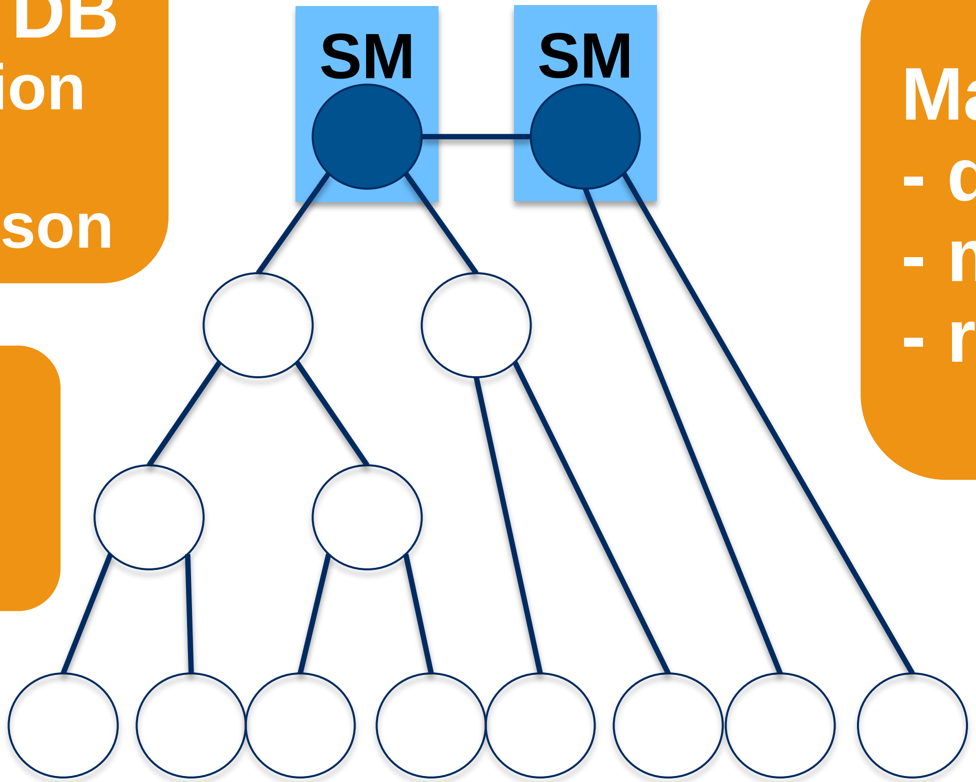
Data Transfer Size in Bytes	Elapsed Time
0.5 GB	0.669 seconds
1.0 GB	1.342 seconds

Core Layer

Core found at SM LID

raw SM DB → SSA DB extraction and comparison

Nodes join SSA tree



Manage SSA group
- distribution control
- monitoring
- rebalancing

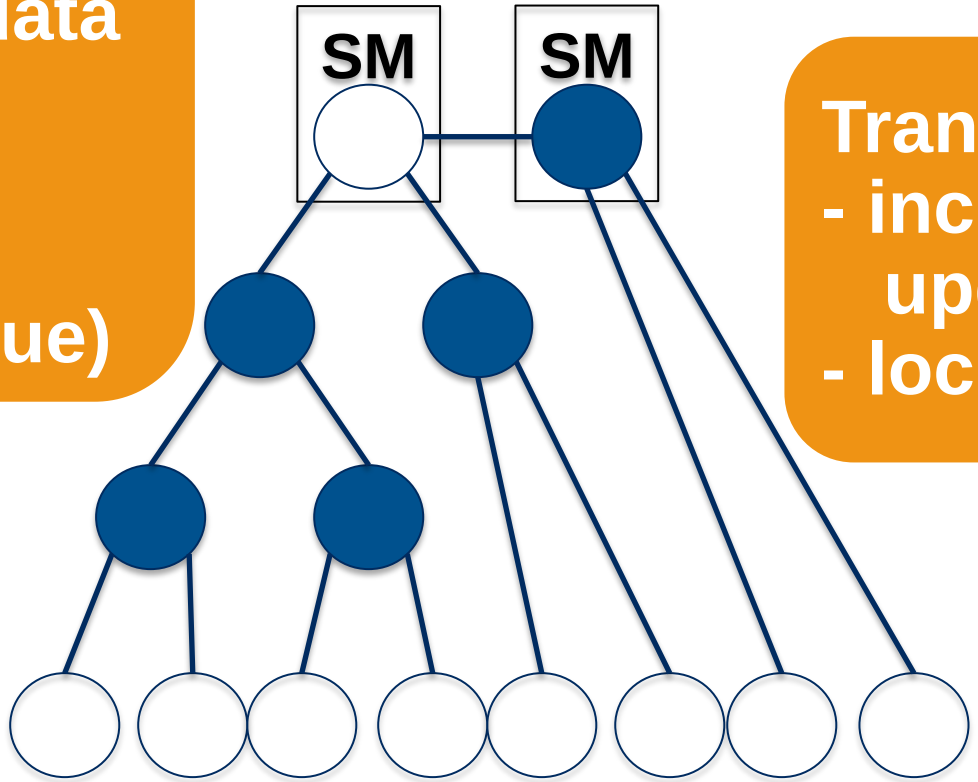
Core Performance

- Initial subnet up for ~20K nodes fabric
  - Extraction: 0.228 sec
  - Comparison: 0.599 sec
- SUBNET UP after no change in fabric
  - Extraction: 0.152 sec
  - Comparison: 0.100 sec
- SUBNET UP after single switch unlink and relink
  - Extraction: 0.190 sec
  - Comparison: 0.865 sec
- Measurements above on Intel(R) Xeon(R) CPU E5335 @ 2.00GHz 8 cores & 16G RAM

Distribution Layer

Distributes SSA DB
- relational data model
- data versioning (epoch value)

Data agnostic

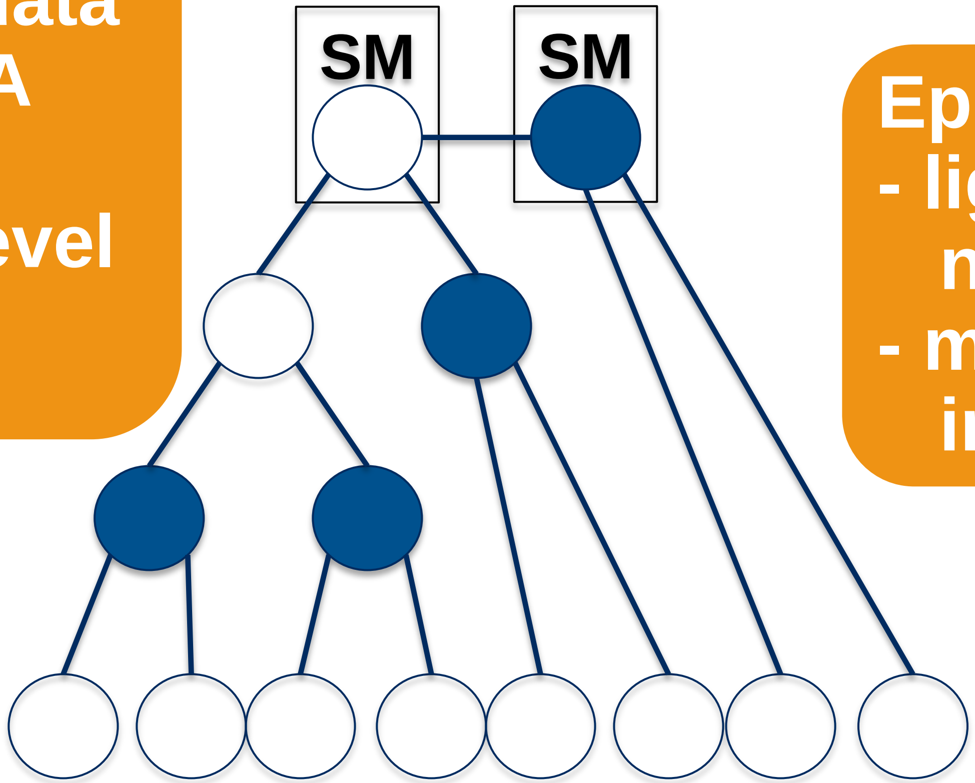


Transaction log
- incremental updates
- lockless

Access Layer

Formats data
- select SA queries
- higher-level queries

Data aware



Epoch value
- lightweight notification
- minimal job impact

Access Layer Notes

- Calculates SMDB into PRDB on per consumer basis
  - Multicore/CPU computation
- Only updates epoch if PRDB for that consumer has changed

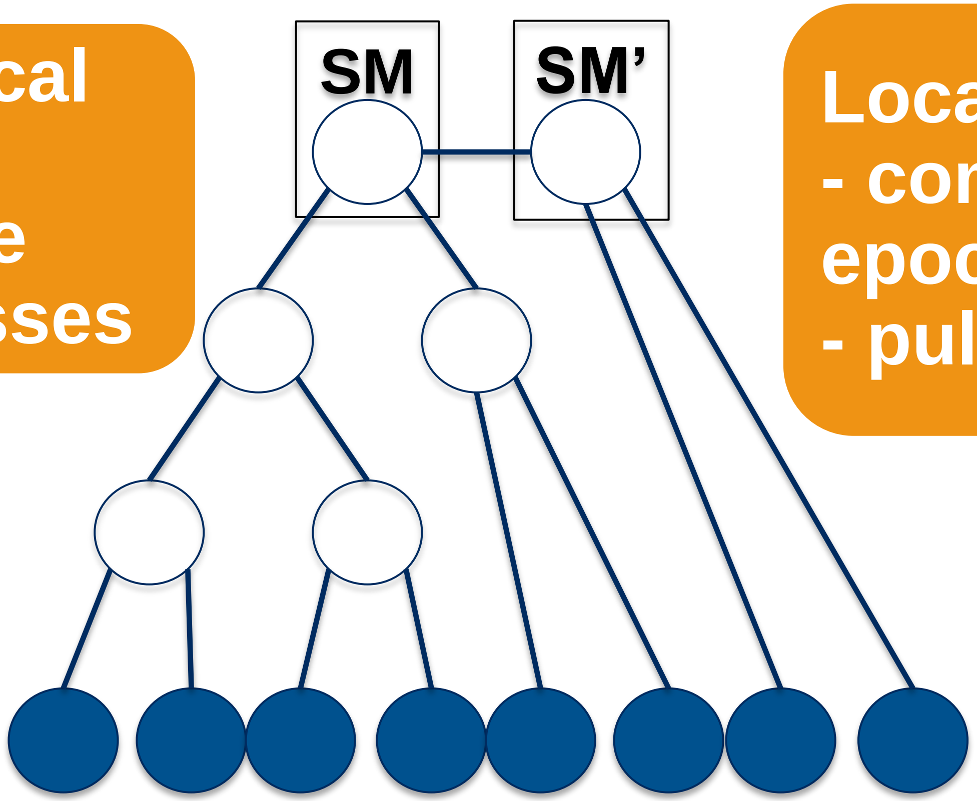
Access Layer Measurements/Future Improvement(s)

- Half world (HW) PR calculations for 10K node simulated subnet
- Using GUID buckets/core approach, parallelizing HW PR calculation works ~16 times faster on 16 core CPU
  - Single threaded takes 8 min 30 sec for all nodes
  - Multi threaded (thread per core) takes 33 seconds
  - Parallelization will be less than linear with CPU cores
- Future Improvement(s)
  - One HW path record per leaf switch used for all the hosts that are attached to the same leaf switch

Compute Nodes (Consumer/ACM)

Integrated with IB ACM
- via librdmacm

Publish local data
- hostname
- IP addresses

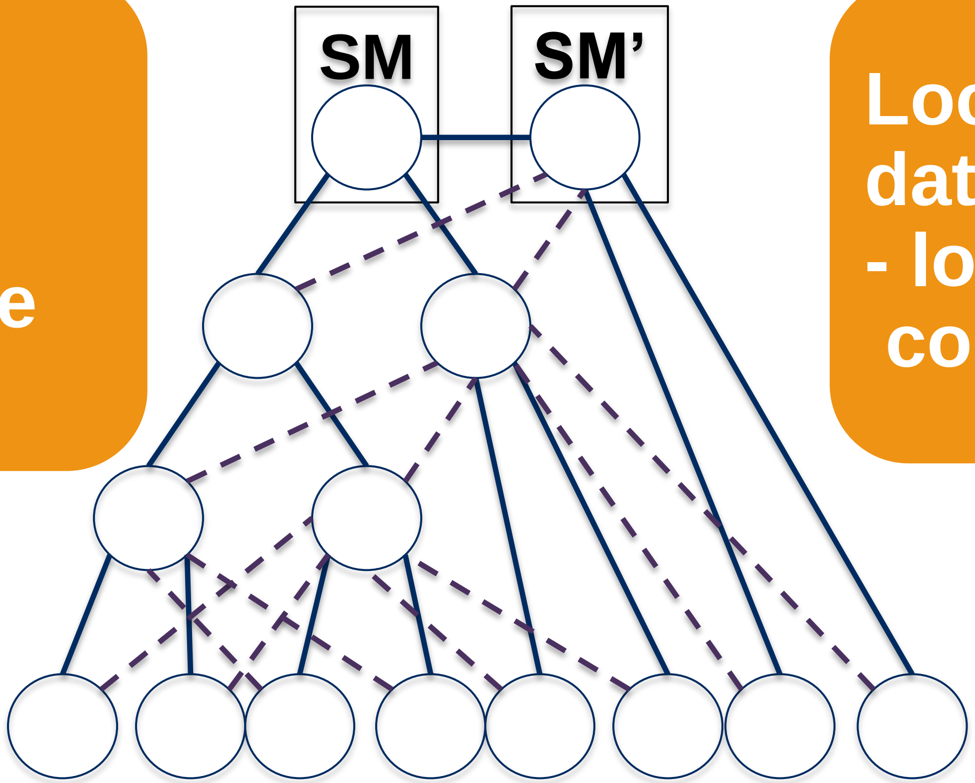


Localized cache
- compares epoch
- pull updates

Reliability

Primary and backup parents

Error reporting - parent notifies core of error



Local databases - log files for consistency

IP Address Support

- Like DHCP in large subnets, hostnames and IP addresses are administered in advance
- SSA functions as a “poor man’s” DNS service
- IP address/hostname file at all core nodes
  - Handle file change
- Enhance SSA DB with IP address and hostname information
  - Update SMDB and PRDB metadata to include 3 new tables
    - Hostname, IPv4, and IPv6
      - Additional flag that says if the data was changed

Kernel ARP Cache

- Approach
  - ACM IPv4 cache is used to make static (permanent) ARP entries in kernel
  - Similarly, IPv6 cache can be used for neighbor entries
- Assumption
  - ARP cache is configured appropriately to hold all needed entries
- For IP addresses to be able to populate ARP cache, the QPN + flags is needed
  - Flags byte from RFC 4755: |RC|UC| 0| 0| 0| 0| 0| 0|
  - If QPN omitted, entry can not be put into kernel ARP cache
- Use netlink routing socket which already supports the needed operations
  - Add/Delete/Get neighbor for both IPv4 and IPv6
  - Only program neighbor entries with QPN != 0

Kernel PathRecords

- Kernel changes for Path Records per ULP via netlink API
  - Ideally, PR cache in kernel should be shared across ULPs
    - IPoIB first ULP (already supports some netlink operations)
  - ULPs synchronize on user space PathRecord cache netlink queries to ACM

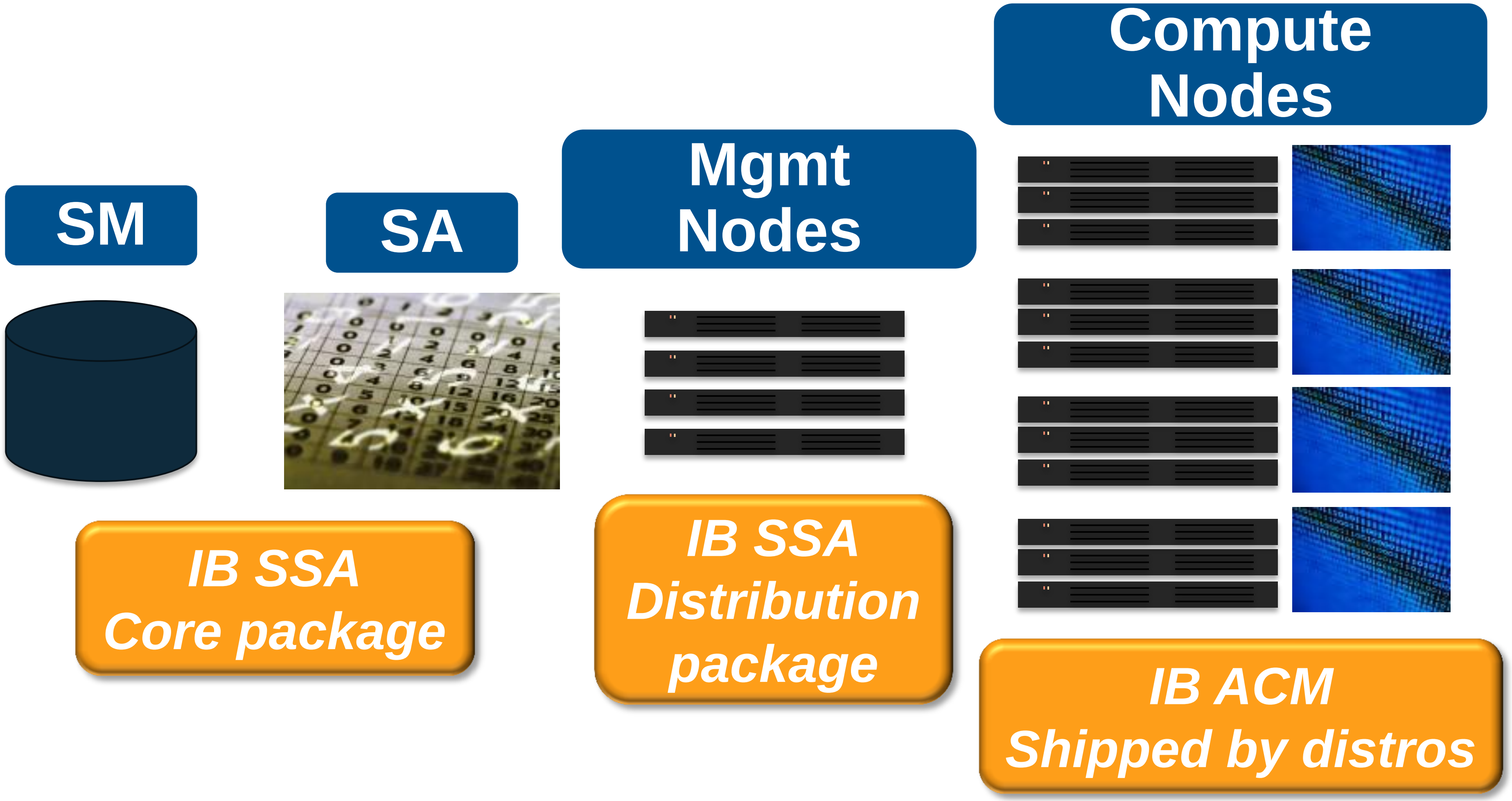
System Requirements

- AF_IB capable kernel
  - 3.11 and beyond
    - SLES 12.0 is 3.12 based
    - Ubuntu 12.01.1 (3.12.0-031200-generic)
    - Ubuntu 14.04 is 3.13 based
    - Fedora Core (Rawhide, FC19 or later)
    - OpenSuSE 13.2 uses 3.16 going for 3.17
    - Note that both RHEL 7.1 and RHEL 7.0 use 3.10 so these do not support SSA
- librdmacm with AF_IB and keepalive support
  - 1.0.20 release
- libibverbs 1.1.8
  - libmlx4 1.0.6
  - libmlx5 1.0.2
- libibumad 1.3.10.2
- OpenSM
  - 3.3.17 release or beyond
  - 3.3.19 release of beyond if running PerfMgr

OpenMPI

- RDMA CM AF_IB connector contributed
  - Part of 1.9 release
- Topology support
  - Future

Deployment



SM

SA

Mgmt Nodes

Compute Nodes

IB SSA Core package

IB SSA Distribution package

IB ACM Shipped by distros

Initial Releases

- Path Record Support
  - Upstream - January
  - MOFED 3.0 - May
- IP Address Support
  - September
- Current Limitations
  - Only x86_64 processor architecture has been tested/qualified
  - Only single P_Key (full default partition - 0xFFFF) currently supported
  - QoS routing and policy
  - Virtualization (alias GUIDs)

Summary

- A scalable, distributed SA
- Works with existing apps with minor modification
- Fault tolerant

Thank You

Mellanox® TECHNOLOGIES

Connect. Accelerate. Outperform.™