They Don't Know What They Don't Know

Is the recently released NonStop In-Memory Cache product good enough for HPC?

Dean E. Malone

August 20, 2017 - Rev 1.5

HPE's overall message to the NonStop community has been positive, as they continue to invest in NonStop. Over the past couple of years we have seen how they have moved quickly to support industry-standard chips and interconnect. For many of us, this was long overdue. Sticking with Itanium and ServerNet, while understandable from a business perspective, left NonStop isolated and perceived as proprietary.

Support for x86 and InfiniBand just happen to be the first steps towards embracing more standards and becoming even more of a legitimate open platform; better able to participate in the infrastructure upgrades that enterprises are pursuing today. X86 and IB started NonStop down this road, but with the addition of virtualized NonStop and support for commercial off-the-shelf hardware, together with RoCE for interconnect pathways, HPE continues to keep NonStop relevant in today's world of hybrid IT.

But this isn't to say that there is not a lot more that can be done, or that all the improvements and capabilities that NonStop customers need, have been addressed. That door must always remain open to new ideas and new implementations if this platform is to become perceived as important to the broad industry. Arguably, it is not there yet. Witness the fact that so few customers are utilizing these advanced capabilities. You have to ask yourself why. Even Richard Buckle recently explored this thought when he recently posted:

> *So, let me cut to the chase – I have as yet to come across any enterprise that have gone down this path or talked to data center managers tackling the finer details of having mixes of traditional and cloud environments supporting anything apart from pilots and prototypes. So, has something gone awfully wrong and are the reports coming from industry analysts overstating reality?*

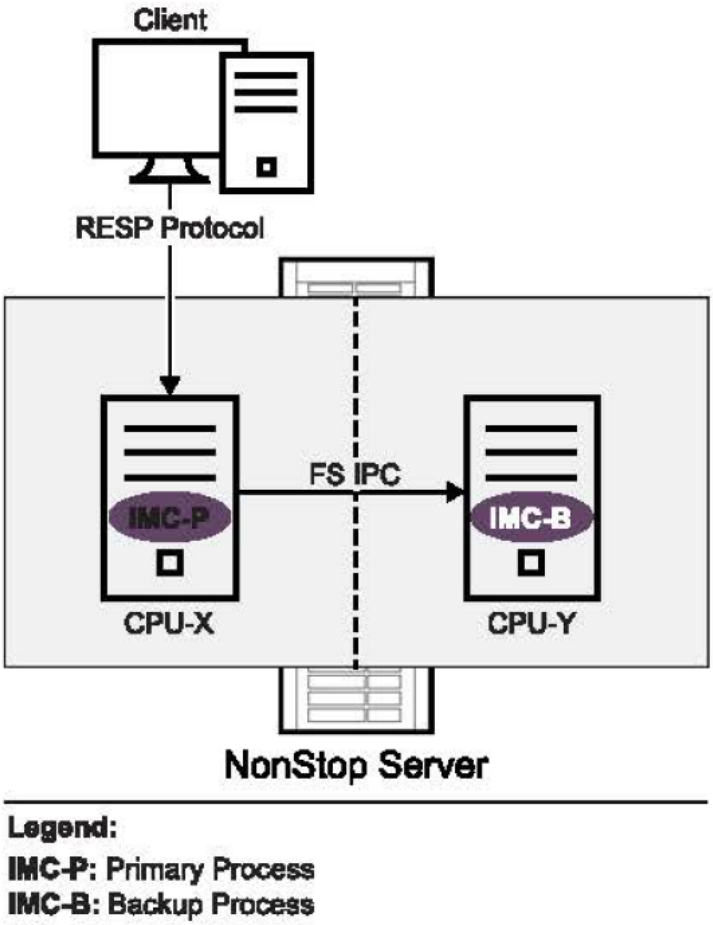
Personally, I think the reason is that there are no robust frameworks that will make it easy to build new and sophisticated applications with, to leverage RDMA in a hybrid computing environment. That spells RISK.

One idea that has been percolating with me for four years now is how to forge a beachhead for business computing into the world of HPC – high performance computing. When I attended the MVAPICH Users Group (MUG) convention, hosted by Ohio State University in 2016, there was an open microphone session at which I asked the question, "Why has the HPC community failed to date to make inroads into the world of business computing?" The panel chair answered by telling us that in 2014, there was a meeting of CEOs and HPC platform vendors who were gathered to consider exactly this. After the presentations, the consensus that those executives offered is that HPC is not robust enough to meet the high availability, security and transaction-based requirements that business requires. It seems to me that if someone could offer a framework which i) supports most - if not all - the most used client and server platforms, ii) can integrate with HPC, iii) provides distributed shared memory transactional rollback/commit semantics, iv) can secure it from an authorization *and* authentication perspective, and v) could be easily managed from an operational perspective, such a product would be a big winner. Think of what MQ Series did with persistent queue-based memory in a distributed computing environment, then multiply that success by an order of magnitude. That's what I see as imminently achievable and what I aspire to build.

What does NonStop and, more broadly HPE, have to offer that can play in this space? HPE recently introduced a distributed shared memory product for NonStop that – like our own XIPC – is fault tolerant. The product is NonStop In-Memory Cache (IMC) and it works on both NonStop Itanium and x86 servers. I also compare IMC to what our own XIPC product can do. I just finished reading the manual, and came to several conclusions.

What I Like

The following picture from the IMC manual depicts its basic architecture:



Here are the things I like:

1. It is a 64 bit application. Thirty-two bit address limitation is so passé.
2. Memory can be accessed seamlessly via other platforms over standard TCP/IP sockets.
3. Abstract data structures (i.e. Redis data types and abstractions) are supported.
4. It will integrate seamlessly with any Redis-based application and adds the valuable dimension of fault-tolerant shared memory.
5. Shared memory can be spooled to a file as part of an orderly shutdown so that upon restart, it can be quickly restored.
6. One can use easily enabled/disabled MEASURE to glean performance metrics.

What I Don't Like

Here are the things I don't like about IMC:

1. It is Guardian IPM I/O bound. It is two orders of magnitude slower than can be achieved by using InfiniBand OFED RDMA instead.
2. When shared memory can be updated by any one of several participant processes, it is ***critical*** that updates are guarded by a mutex (i.e. pair of semaphores – one to acquire access and one to protect the work); otherwise the application is not thread safe. This implementation and therefore applications using it will not be thread safe.
3. When a client application sends what HPE calls a CUD (i.e. change/update/delete), all other clients are blocked until that operation completes – even if the checkpoints take multiple checkpoint writes. The manual says as much. With Guardian tags and the ability to map each reply buffer to its own request tag, IMC does not appear to be making use of them. It appears to be a single-threaded server.
4. If the IMC primary process fails, all the pending sockets do an ECONNRESET and are forced to reconnect because the TCP sockets are not checkpointed and sync IDs are not tracked. Checkpointing has been reduced to a single IPM checkpoint – but at a steep price. This is not an AL4 compliant implementation. Had this been done with Guardian sockets, the remote platform would not even see the process failure because the backup would take over and parallel TCP/IP would hide the fault from the remote client.
5. It has a window of vulnerability whereby the backup gets updated and the primary fails before the remote client gets I/O completion; the backup takes over with changed state but the client thinks the I/O failed.

IMC is Guardian IPM I/O Bound

Here are the ping-pong IPM metrics I collected on an NS7 server in 2015:

Prosess Pairs	Itanium avg elapsed time (sec)	Aggregate per second	Per-message Latency (ns)	Avg per second	Degradation	cpu 0/1	cpu 1/2	cpu 2/3	cpu 3/0	NonStop X Avg Elapsed time (sec)
one pair	452.60	11,047.28	90,520.00	11,047.28	100.00%			11,047.28		94
4 pairs	480.75	41,612.46	96,125.05	10,403.12	94.17%	10,615.71	10,482.18	10,351.97	10,162.60	97.25
8 pairs	577.75	69,367.35	115,328.03	8,670.92	78.49%	8,580.77	8,701.12	8,758.74	8,643.05	108.88

Aggregate per second	Per-message Latency (ns)	Avg per second	Degradation	cpu 0/1	cpu 1/2	cpu 2/3	cpu 3/0	Performance Delta	Note: All results are per second; 5M messages
53,191.49	18,800.00	53,191.49	100.00%			53,191.49		4.814893617	10% CPU busy
205,659.58	19,449.62	51,414.90	96.66%	51,020.41	51,546.39	51,546.39	51,546.39	4.942259679	20% CPU busy
367,964.92	21,741.20	45,995.62	86.47%	47,657.95	46,957.67	44,515.67	44,851.17	5.304583781	40% CPU busy

The first group of results are for Itanium and the second are for x86 – about 5 times better.

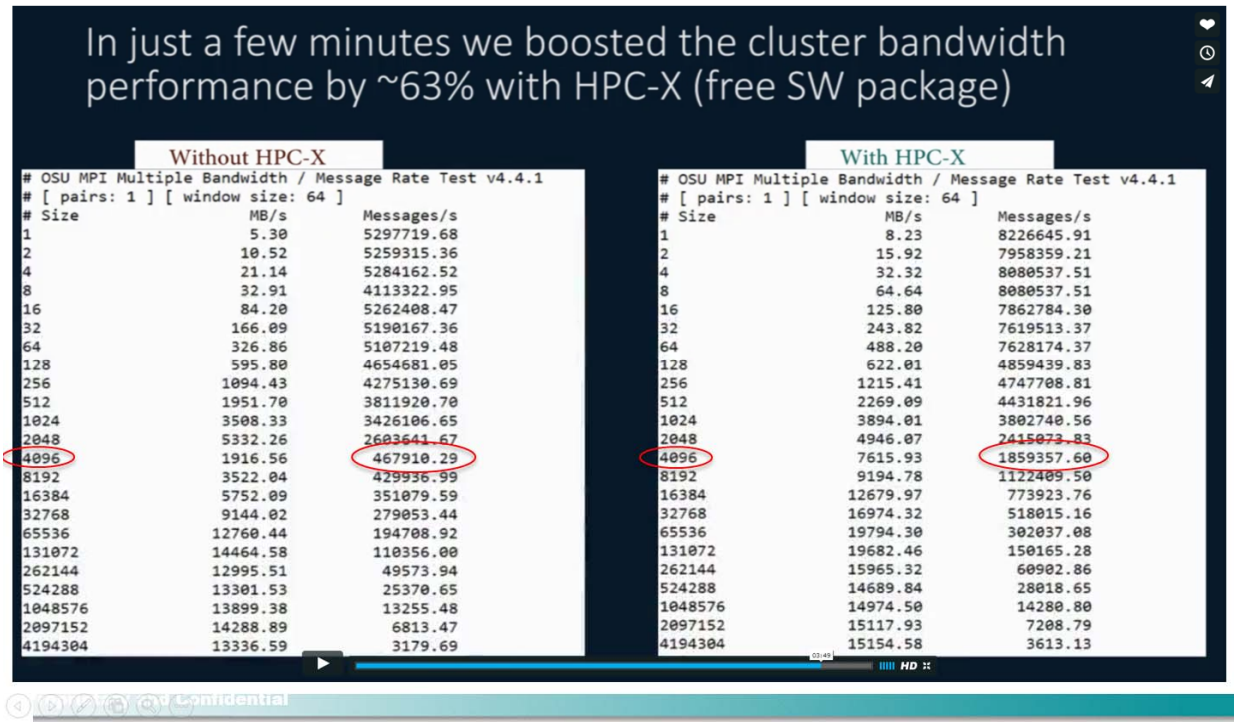
Round-trip IPM latency is somewhere between 18,000 and 21,000 ns (billionths), depending on how busy the server is. This is at the heart of why NS7 is only five times faster than Itanium, when RDMA memory I/O is 8 times faster and would make x86 40 times faster than Itanium if used instead. This is the NonStop x86 bottleneck. What does this have to do with IMC? Lots! Per the IMC manual:

> In the default configuration, IMC runs as a NonStop OSS process pair. All the clients connect to the primary process. ***The primary process performs checkpoint operations on the backup process.*** This way the primary and backup process are always in synchronization with each other, and thus ensuring data consistency during a process failure.

Those checkpoint messages occur via Guardian IPM as *writes* to the backup process $RECEIVE queue. IPM messages are constrained to 57,344 bytes per checkpoint message, so buffers bigger than this need to be checkpointed in sequential segments – each segment being completed with its own REPLYX. Since a $RECEIVE queue only holds 15 pending messages, no more than a dozen pending checkpoints could be sent in a no-wait fashion. What impact does this have on throughput? Again from the IMC manual:

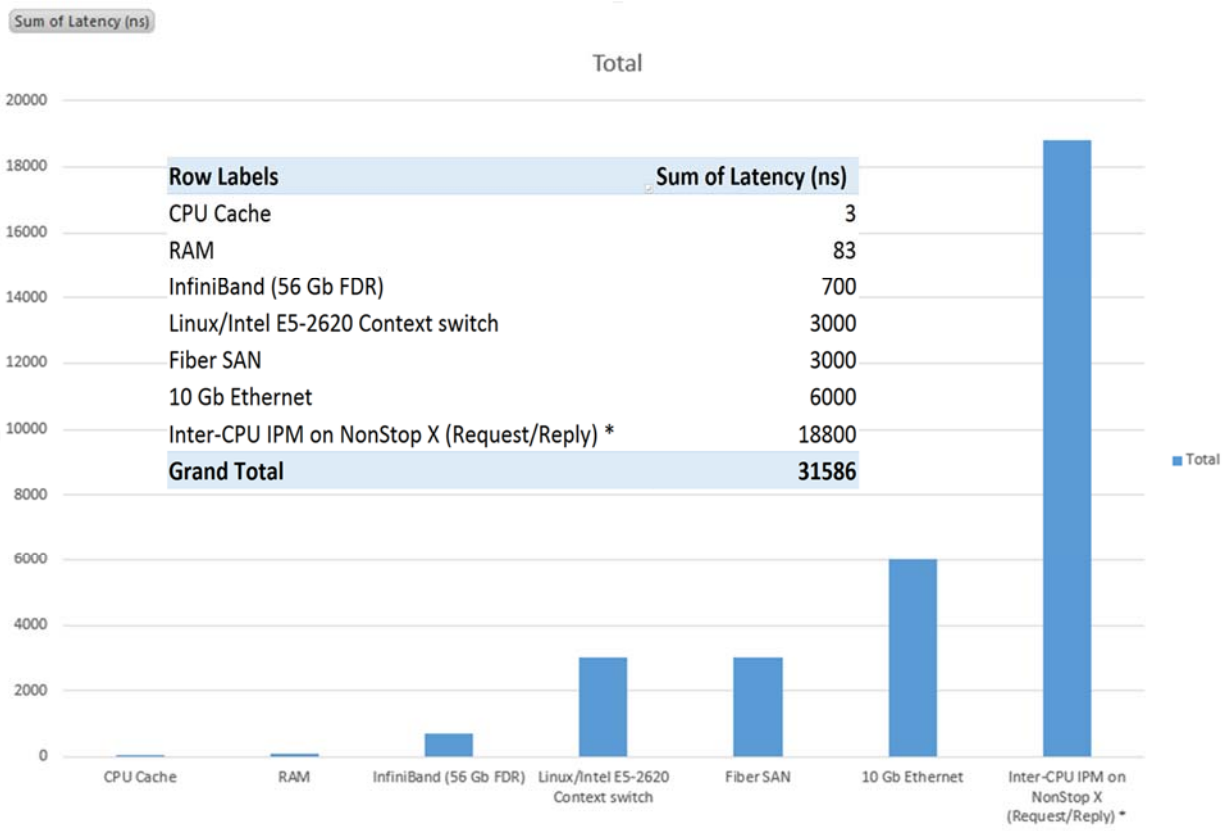
> The duration in IMC is considerably small ***(tens of ms) for data of size 100MB*** to as much as 20 seconds for data of size 15 GB.

Per the metrics I gathered above, I am skeptical of this claim. But let's assume this is correct, and consider what performance would look like if OFED RDMA verbs were used to checkpoint instead. I need look no further than the table Mellanox showed at MUG-2016:



By the metrics per the slower left-side table that does not use the faster HPC-X, it would take twenty-four 4,194,304-byte writes to move 100 MB, so that means an RDMA checkpoint would be at least 132 times faster. But… this checkpoint could be achieved with a *single* RDMA write, which would likely be more than twice as fast; resulting in at least 200 times faster checkpoints. Fifteen GB would take a 10th of a second – not 20 seconds.

For the record, I want to correct myself from the article I wrote for the April 2017 issue of NonStop Insider per the following table. I quoted a scale of millionths of a second but the correct scale is billionths - a nanosecond. The graph is correct but the units reported is wrong. If Guardian IPM took 18,000 µs, that would be terrible. It's not *that* bad. Thanks to Bruce Holenstein for pointing this out.



Row Labels	Sum of Latency (ns)
CPU Cache	3
RAM	83
InfiniBand (56 Gb FDR)	700
Linux/Intel E5-2620 Context switch	3000
Fiber SAN	3000
10 Gb Ethernet	6000
Inter-CPU IPM on NonStop X (Request/Reply) *	18800
Grand Total	**31586**

IMC Is Not Thread-safe

Without the ability to protect memory updates with a mutex, it is an indisputable fact that it is impossible to ensure atomic updates. Furthermore, that mutex would need to be a platform-independent implementation to work in a hybrid environment.

XIPC has been used for over two decades for real-time processing on hundreds of thousands of servers, doing exactly this. As will be elaborated upon shortly, the NonStop XIPC implementation of semaphores is completely fault tolerant. This has already been demonstrated to both HPE and several NonStop vendors. *Any* supported platform can access those semaphores. At this time, the following are 64-bit implementations of XIPC that are entirely thread safe: HPUX, Linux, AIX,

Solaris and Windows. They can interact with both 32 and 64 bit implementations of XIPC with the only limitation being the 32-bit address space constraint of the former.

IMC Is a Blocking Implementation

Any time a client request is blocked by another pending request, this is evidence of a single-threaded implementation. With Guardian tags, there is absolutely no reason – other than additional program complexity – why IMC could not have been implemented in a multi-threaded manner. There are two statements in the IMC manual that suggests this is true:

> Primary process handles all the client requests and the clients request completion status is communicated only upon successful completion of data checkpoint to the backup process.

And:

> Also, it synchronizes all the data to the backup process before processing any new requests.

XIPC does not block on checkpoints to the backup. In fact, it never blocks on any I/O. The way I implemented XIPC, it is completely no-wait and event driven. Every unsolicited client request has a Guardian Tag. A unique $RECEIVE buffer is dynamically allocated for every request, and when a request blocks (e.g. queue read on an empty queue) the context for the request is saved until the blocking operation becomes unblocked. The 32-bit Guardian tag is a dynamically allocated context control block (CCB) linked-list element address that is guaranteed to be unique. When I/O completes, the saved context checkpoint I/O completion tag (i.e. CCB address) is used to immediately obtain the Guardian reply TAG, and the $RECEIVE reply buffer associated with it is used to construct and REPLYX. This is a completely thread-safe multi-threaded implementation.

IMC is Not AL4 Capable

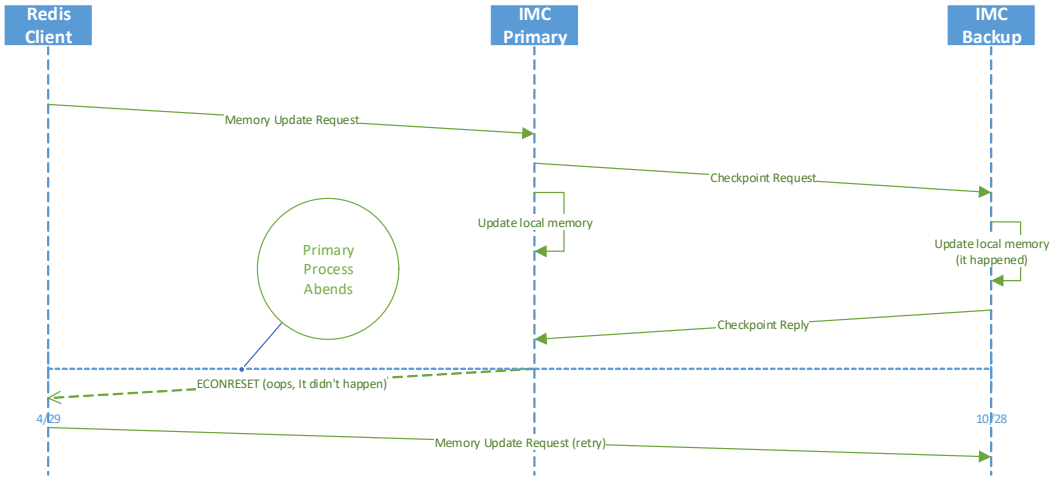
HPE has made a big point of boasting that NonStop Servers are AL4 compliant. That is a big deal. The HPE products that run on it should also be AL4-capable. What evidence do I have that IMC is not AL4 capable? It's again in their own IMC manual:

> Upon an abnormal termination of primary process, all the client connections to the server are reset and the clients have to reconnect to the server.

In 2015, I demonstrated the XIPC AL4 capability to do exactly this to the HPE EMEA architects team. The goal was to demonstrate that fault-tolerant semaphores could reside on a NonStop which would ensure reliable delivery of boxcars of data (i.e. 100 queued messages at a time) to Vertica. We initiated the delivery of 10,000 messages in 100-message boxcars from NonStop message queues to a HPUX server. Every time the HPUX server set the semaphore, 100 messages would be sent to it. While these messages were in flight, we did a DIVER of the primary process's processor. The HPUX server hesitated for an almost indiscernible amount of time but the TCP/IP socket did not reset, the semaphore was intact and all 10,000 messages were successfully delivered with no loss or duplication.

IMC has a Window of Vulnerability

The following time-sequence UML diagram says it all. Note the *wrong* result if failure occurs at the time noted at the dotted blue line – a window of vulnerability:



Final Thoughts

Though I think this is an interesting product, IMC is already obsolete technology that does not come even close to utilizing the X Series platform's advanced InfiniBand capabilities. I think something *much* better can be built and indeed, I am *much* closer to the finish line of this goal than anything I have seen offered by NED or HPE.

Meg Whitman did a big dog-and-pony show the week of May 14th called *HPE Unveils Computer Built for Big Data Era* and it is the latest HPE plug for The Machine with 160 terabytes of memory. Did you see it? http://tinyurl.com/y8otdkyg. This *is* a very impressive Machine. Be assured though that this address space is being referenced by OFED RDMA – even if the trailer didn't say so. Given that this is where HPE as an enterprise is headed, why isn't NED beating the same drum? Furthermore, it is obvious to any competent architect that The Machine will not be able to survive a single point of failure because it is not built with redundancy, like a NonStop is. This is a niche that NED must capitalize on if NonStop is to survive another 40 years.

My view, as an architect, is that unless I am porting code that is already using this Redis shared-memory framework, I would not build an application with IMC for the simple reasons that: 1) it is way too SLOW, 2) it has no synchronization mechanisms for controlling concurrency of access in a distributed computing environment and 3) it has a window of vulnerability.

One of the things several NonStop partners and even a few HPE "insiders" have asked me is, "Why is HPE not all over this?" Yes, I also wonder why HPE will not get behind XIPC as its hybrid RDMA framework. After four years of dialogue, is IMC the best NED could come up with? What HPE is doing with The Machine and what NED is doing with NonStop are out of step. Then it hit me. Maybe they just don't know what they don't know. Buy it or build it; but IMC is not *it*. Contact me at dean@caleb-ltd.com to share your thoughts or visit my web site. www.caleb-ltd.com