High Performance Spaceflight Computing (HPSC)
Next-Generation Space Processor (NGSP)
A Joint Investment of NASA and AFRL

Richard Doyle, Raphael Some
Jet Propulsion Laboratory, California Institute of Technology

Wesley Powell
NASA Goddard Space Flight Center

Gabriel Mounce
Air Force Research Laboratory

Montgomery Goforth
NASA Johnson Space Center

Stephen Horan
NASA Langley Research Center

Michael Lowry
NASA Ames Research Center

Abstract

Spaceflight computing is a natural technology multiplier for space missions. NASA recently developed a set of use cases for future flight computing capability and derived a set of reference requirements. NASA evaluated several computing architectures and concluded that a multicore approach would provide the greatest value to realize both a significant computational advance and flexible architectural support for the demands of space missions. AFRL had been independently planning for a flight computing investment, and the two agencies have entered into a partnership to develop and evaluate hardware architecture designs for a future flight computing system solution. The current phase of the joint investment is focusing on retiring technical risk. A subsequent phase would develop a board-level flight computing system solution for validation and first mission use.

Introduction

Spaceflight computing can be aptly viewed as a "technology multiplier" for space missions, in that advances in flight computing capabilty will provide the basis for broad improvements in flight functions and capabilities, and will enable new mission scenarios, increasing science and exploration return.

In 2012, the NASA Game Changing Development Program (GCDP), residing in the NASA Space Technology Mission Directorate (STMD), commissioned a High Performance Spaceflight Computing (HPSC) formulation activity to determine the value of an investment in a next-generation flight computing system. A multi-center NASA team was established to sharpen understanding of the gap between the current state of the practice in flight computing and the actual future flight computing needs of NASA missions. This team posed the question: "What are the paradigm shifting NASA space-based applications that drive flight computing?" A series of workshops was conducted with personnel from NASA Johnson Space Center (JSC), NASA Goddard Space Flight Center (GSFC) and the Jet Propulsion Laboratory (JPL). These scientists, engineers and mission designers identified high priority functions, capabilities and mission scenarios from the NASA future mission set, and identfied use cases that required or would take full advantage of a high performance spaceflight computing capability. Both robotic and human spaceflight mission applications were examined.

To answer the related question: "What are the requirements imposed on flight computing by these applications?" NASA system engineers and mission personnel characterized for each use case: the nature of the computing, the environment, the criticality of the application, the system constraints, and the computing system and processor chip requirements.

Analysis of the set of use cases and derived requirements led to the insight that the drivers for future flight computing fall into three broad categories: 1) hard real-time calculations (such as vision-based algorithms for entry, descent and landing), 2) high throughput instrument data processing (such as for hyper-spectral and synthetic aperture radar – SAR), and 3) model-based reasoning techniques, such as for AI-based mission planning and fault management). The takeaway was that there would not likely be a one-size-fits-all architectural solution.

The HPSC formulation team next evaluated extant and emerging computing architectures to determine which architectural concept would provide the most investment value to address the identified future NASA flight computing requirements. Several architecture were examined, both COTS (commercial off-the-shelf) -based, and, where available, space-qualified or space-qualifiable versions: Reconfigurable Computing, Graphic Processing Units (GPUs), Multi-Core, specialized processors (e.g., DSP – Digital Signal Procsssing), as well as hybrid approaches. The study identified rad-hard general-purpose multi-core as the highest value architectural approach for NASA's needs – not only addressing the range of flight computing requirements derived from the use cases, but providing certain architectural advantages in a flight computing system beyond the computational performance advance.

As NASA's HPSC activity proceeded, the Air Force Research Lab (AFRL) and NASA identified significant requirements overlap and common interest in future flight computing. An agency-level partnership emerged, which is manifesting through a joint investment in a Next Generation Space Processor (NGSP).

In 2013, under an initial risk-retirement phase, termed the Innovation Phase, AFRL and NASA issued awards via a joint solicitation to three companies: BAE, Boeing and Honeywell. These contractors are developing hardware architecture

designs for the targeted future flight computing system based on a multi-core architecture. NASA is also developing a set of benchmarks consistent with the broad set of NASA applications. Some benchmarks will capture performance needs but others are for the purpose of evaluating certain system-level properties of the operational details of vendor-offered designs, such as support for energy management and fault tolerance. These benchmarks are targeted for flight computing, and as such, are different than the more familiar computing benchmarks that are available in the ground-based supercomputing community.

The hardware architecture designs will be developed and evaluated in 2014. Assuming a viable departure point has been achieved with sufficient risk retirement, in 2015 a three-year Development Phase will follow, to culminate in a board-level flight computing system product, incorporating a multi-core chip, with integrated real-time operating system, flight software development environment, and NASA-developed middleware to fully exploit multi-core architectural features for the benefit of the future spaceflight applications.

NASA and AFRL intend to continue in partnership through the Development Phase, and are seeking additional programmatic partners for this national-level capability investment.

Use Cases

To identify use cases for a next-generation flight computing capability, a series of workshops was held with scientists, engineers and mission designers from NASA Johnson Space Center (JSC), NASA Goddard Space Flight Center (GSFC) and the Jet Propulsion Laboratory (JPL). NASA Ames Reseach Center (ARC) and NASA Kennedy Space Center (KSC) also supported the workshops. Both robotic and human spaceflight mission applications were examined.

The workshops identified a range of future NASA flight applications for significantly improved flight computing (defined as 100X compututational perfromance advance over the RAD750), projected through the 2025 timeframe. Nineteen generic applications were identified that required or would take full advantage of significantly higher performance computing than available with current or planned space-qualified computers.

Table 1 lists the identified use cases, organized by the two primary NASA Mission Directorates: the Human Exploration Mission Operations Diretorate (HEOMD) and the Science Mission Directorate (SMD).

Table 1 — Future NASA Mission Use Cases for
High Performance Spaceflight Computing

HEOMD Use Cases	**SMD Use Cases**
Cloud Services	Extreme Terrain Landing
Advanced Vehicle Health Management	Close Proximity Operations/Formation Flying
Crew Knowledge Augmentation	Fast Traverse
Improved Displays and Controls	New Surface Mobility Methods
Augmented Reality for Recognition and Cataloging	Imaging Spectrometers
Tele-Presence	Radar
Automated Guidance, Navigation and Control	**Low Latency Products for Disaster Response**
Human Movement Assist	Space Weather
Autonomous & Tele-Robotic Construction	Autonomous Mission Planning
	Immersive Environments - Science Operations/Outreach

Space limitations preclude offering a description of the full set of NASA use cases [1,2,4,7], but to provide a sampling of the range of space-based capabilities enabled or advanced by HPSC capability, the use cases accented in **bold** above are expanded below. However, a straightforward conclusion is that HPSC will indeed be game changing because the capability will be needed for many future NASA missions, and will enable new and dramatic flight applications.

Advanced Vehicle Health Management

Integrated Vehicle Health Management (IVHM) or Integrated System Health Management (ISHM) enables better management of vehicle health. This objective is achieved through correct use of reliable sensing to monitor component health combined with vehicle/system usage, knowledge of likely future events, and diagnostic and prognostic reasoning to optimize vehicle safety, reliability, availability, and maintenance efforts.

Attempts have been made by NASA to address IVHM, but they have not met with full success; the amount of sensor data and processing power available has never allowed an adequate implementation.

Providing this processing power will allow the monitoring of a greater number of sensors at higher frequencies along with the comparison of this data over time and between areas of the vehicle, all of which will significantly enhance failure detection. Additional processing power will also allow the implementation of more

sophisticated isolation and recovery functions, and the implementation of algorithms that predict impending failures, thus enabling preventative maintenance.

A comprehensive implementation of IVHM will be extremely important to the success of future long duration exploration missions, including crew-tended Cis-Lunar missions, which may involve a significant amount of un-crewed operations. IVHM will allow optimum use of crew time for maintenance by allowing the risk of failure to be balanced against the resources required for scheduled maintenance. Ultimately, IVHM capabilities will evolve to the implementation of a "Digital Twin", per the NASA Office of the Chief Technologist (OCT) Strategic Roadmap TA11 for Modeling, Simulation, Information Technology and Processing.

Low-Latency Products for Disaster Response

Successes with science event detection and response on Earth-observing spacecraft, notably the Autonomous Sciencecraft capability that has been operational on the Earth-Observing One (EO-1) spacecraft since 2004, led to further thinking about whether event detection and response could be generalized across multiple platforms: flight, ground and possibly even air-based. An event of interest might be detected initially on any given platform, but the optimal follow-on observational response might best occur on a different platform with a better view or sensor.

Improved onboard computing capacity can greatly assist such objectives by: 1) enabling rapid turnaround of products for disaster response (hours/minutes instead of weeks), 2) automatic cueing of additional assets such as other space platforms or UAVs, not currently feasible due to time delays, 3) direct broadcast of evolving data products in the field for situational awareness and decision support, and 4) delivery of real-time high-definition video.

The specific onboard data processing capabilities required by these scenarios include fast spatial/spectral sub-setting, geo-rectification, atmospheric correction, rapid broadcasts in the event of library matches, along with the ongoing challenges of processing data streams approaching 1 Tb/sec (as is the case with the HyspIRI hyperspectral instruments). Such intense onboard data processing would likely impact other aspects of the flight system; there would need to be sufficient memory and storage for both processing and archiving. Downlink capacity, although greatly ameliorated by onboard processing, would still need to be sufficient to support direct broadcast objectives.

Derived Requirements

To derive requirements for future NASA flight-based computing, follow-on discussions were held with NASA personnel beyond the engineers, scientists and mission designers involved in the development of the use cases. The HPSC team derived computing system and processor requirements for each use case by

characterizing the required computing, the environment, the criticality of the application, and the system constraints [3,5,6].

The requirements template developed for this purpose is shown here:

- *Space Environment(s)*
  - Radiation environment at the time of application; e.g., geosynchronous (GEO), low-Earth orbit (LEO), deep space.

- *Spacecraft Power Environment(s) / Constraint(s)*
  - Available power for avionics and computing, e.g., small spacecraft or rover with limited power availability (6 Watts processor power, 10-15 Watts computer power), medium sized spacecraft (7-12 Watts processor power, 15-30 Watts computer power), or large spacecraft with large power budget (>12-25 Watts processor power, >30-100 Watts computer power)

- *Criticality/Fault Tolerance*
  - Is this application life or mission critical, must it operate through faults, can it tolerate errors if detected?

- *Real-Time*
  - Does the application have a hard real-time deadline; if so, what is the required timing?

- *Type(s) of Processing*
  - Algorithm kernel(s). Is it primarily e.g., digital signal processing (DSP), data base query, is it parallelizable, is it amenable to a data flow approach?

- *Memory Access Pattern*
  - What is the primary data movement pattern, e.g., does it fetch data from memory once and then flow through a processing chain, or does it access data in a continuous random access pattern, or does it access sequential data in a continuous and regular pattern?

- *Duty Cycle*
  - What is the pattern of execution, e.g., is the application called continuously over a long period of time, is it called once and operate for only a short duration, is the application execution profile spiky and/or unpredictable?

- *Data Rate*
  - What are the I/O and memory access data rates?

Table 2 summarizes the application requirements set. As shown, the application list was augmented beyond the set of use cases with known applications from missions currently under development or in an early study phase.

The table is primarily organized vertically, by application throughput requirement. The first section of the table lists applications requiring one to ten giga-operations per second (GOPS), the second section lists applications requiring ten to fifty GOPS, and the third section lists applications requiring fifty to one hundred GOPS.

Each application is then characterized with respect to its primary processing requirements by the following parameters:

Type of processing: Digital signal processing (DSP), general purpose processing (GP), and parallelizability of algorithms (P). Applications requiring at least 20% of any of these types of computing were so noted.

Mission criticality: Mission critical applications were assumed to require extremely high levels of reliability, and therefore, fault tolerance.

Power: Available power for spacecraft computing (LP=low power, MP=medium power, HP=high power) per the requirements template above.

Radiation environment: Radiation environment drives both the radiation tolerance and fault tolerance requirements of the processing system.

Memory access: R=random, S=sequential

Duty cycle: C=continuous, S=spiky

Memory access rate: In bytes per second

I/O rate: In bits per second

Table 2 – NASA Applications Requirements Summary

Apps Summary	Processing Type			Criticality	Available Spacecraft Power							
	DSP	GP	P	MC	LP	MP	HP	Rad Env	Mem Access	Duty Cycle	Mem Rate	I/O Rate
Throughput = 1-10 GOPS												
Autonomous Mission Planning		X	X	X	X	X	X	All	R	C,S	1GB/S	100Mb/S
Disaster Response	X	X			X			LEO	R,S	C	200,B/S	1Gb/S
Hyspiri	X	X	X			X		LEO	R,S	C	200MB/S	
Throughput = 10-50 GOPS												
Fast Traverse	X	X	X	X	X			Mars	R, S	C	500MB/S	
Extreme Terrain Landing	X	X	X	X	X			Mars, GEO	R, S	C	50MB/S	
Adept		X	X			X		LEO, MEO	R, S	C	375MB	
Optimum Observation	X	X	X		X	X	X	All	R	C	1GB/S	1Gb/S
Space Weather	X		X		X	X	X	All	R	C	125MB/S	500Mb/S
Robotic Servicing	X	X	X	X	X	X	X	GEO	R	C,S	125MB/S	
Cloud Service	X	X	X				X	All	R,S	C,S	6.25GB/S	10Gb/S
Advanced ISHM		X	X		X		X	All	R	C	6.25GB/S	10Gb/S
Autonomous and Telerobotic Construction		X	X	X	X		X	GEO, Mars, Lunar	R	C,S	12.5GB/S	50Gb/S
Througput = 50-100s GOPS												
Hyperspectral Imaging	X	X	X		X	X	X	All	R,S	C,S	nGB/S	
RADAR Science	X	X	X				X	All	R,S	C	1GB/S	
RADAR EDL	X		X	X	X	X	X	All	R,S	S	1GB/S	
Automated GN&C	X	X	X	X			X	All	R,S	S	12.5GB/S	10GbS
Human Movement Assist	X	X	X		X			All	R,S	S	12.5GB/S	50GbS
Crew Knowledge Augmentation		X	X					All	R,S	S	12.5GB/S	10Gb/S
Improved Displays and Controls		X	X	X	X		X	All	R,S	C	12.5BG/S	50Gb/S
Augmented Reality		X	X		X		X	All	R,S	S	12.5GB/S	50Gb/S
Telepresence		X	X		X		X	All	R,S	S	12/5GB/S	50Gb/S

Each of the applications will be utilized in multiple missions and scenarios. As an example, Autonomous Mission Planning appears on small, medium and large spacecraft, in a multiplicity of radiation environments. In at least some cases, this application will be mission critical.

At a higher level, the NASA use cases (both HEOMD and SMD) cluster into three broad categories of computational drivers for future flight computing needs: 1) hard real-time calculations (such as vision-based algorithms for entry, descent and landing), 2) high throughput instrument data processing (such as for hyper-spectral and synthetic aperture radar – SAR), and 3) model-based reasoning techniques, such as for AI-based mission planning and fault management). See Table 3. The takeaway is that there was not likely to be a simple one-size-fits-all solution.

Table 3 – Mapping of the NASA Use Cases to the Computational Driver Categories

Computational Driver Category	**HEOMD Use Case**	**SMD Use Case**
Vision-Based Algorithms with Real-Time Requirements	Crew Knowledge Augmentation	Extreme Terrain Landing
	Improved Displays and Controls	Fast Traverse
	Augmented Reality for Recognition and Cataloging	Immersive Environments - Science Ops/Outreach
	Automated Guidance, Navigation and Control	
	Human Movement Assist	
Model-Based Reasoning Techiques	Advanced Vehicle Health Management	Autonomous Mission Planning
	Autonomous & Tele-Robotic Construction	Close Proximity Operations/Formation Flying
		New Surface Mobility Methods
High Rate Instrument Data PRocessing	Cloud Services	Low Latency Products for Disaster Response
	Tele-Presence	Imaging Spectrometers
		Radar
		Space Weather

Evaluation of Computing Architectures

Having identified use cases and derived an initial set of requirements for a future flight computing capability, NASA's next step in formulation was to survey extant and emerging computing architectures and evaluate them in two ways: 1) for relevance in providing an architectural solution that would address the NASA needs,

and 2) to verify that any remaining technical gaps would be within reach of an investment that could be made from the NASA Game Changing Development Program. [10]

An initial step was to identify the care-abouts, or Key Performance Paramters (KPPs) that would be used to evalaute the computing architectures. The set of KPPs used by NASA for this purpose, with brief descriptions, appear below.

- Computational Performance
  - This KPP provides the basic measure of processor performance
- Fault Tolerance
  - This KPP assesses the use of higher-level design or system techniques to mitigate faults that may result from radiation induced upsets, latent fabrication defects, or other causes
- Radiation Tolerance
  - This KPP measures the degree of inherent resistance of an architecture to radiation induced upsets and degradation
- Power and Energy Management
  - This KPP assesses processor power dissipation as well as the ability to scale power consumption to the level of computational activity needed
- Programmability and Flight Software Applicability
  - The requirement to be assessed under this KPP is whether sufficiently mature tools are available to allow highly reliable software to be developed, debugged, tested and verified cost effectively
- Flight Software Verification and Validation (V&V)
  - This KPP assesses avoidance of issues such as on-chip asynchrony as well as tool support and features that support hardware-in-the-loop flight software debugging
- Interoperability
  - The interoperability KPP evaluates the ability of a given architecture to work with other processing elements
- Extensibility and Evolvability
  - This KPP evaluates a given architecture's ability to support extension, scaling and longevity of usefulness
- Cross-cutting potential across NASA missions
  - This KPP assesses breadth of performance and general architectural support across all or most of the identified use cases and derived requirements
- Non-recurring cost
  - This KPP assesses whether a device can be delivered that satisfies requirements and addresses technical gaps within the programmatic resource envelope expected to be available
- Recurring cost
  - The recurring cost KPP is a coarse estimate of the cost to manufacture a processor based upon the given architecture

Armed with this set of KPPs, the NASA team evaluated the following set of computing architectures:

- General Purpose Multicore – COTS and Rad Hard
- DSP Multicore – COTS and Rad Hard
- Graphics Processor Units (GPU) – COTS and Rad Hard
- Reconfigurable Computers (e.g., FPGAs) – COTS and Rad Hard

Several of the COTS-based architectural candidates (specifically, COTS-based DSP multicore, COTS-based GPUs and COTS-based reconfigurable computing) were eliminated because the gap to address basic NASA needs for fault tolerance and energy management were evaluated to be too great to close under a reasonably scoped investment.

The remaining candidate architectures were evaluated as shown in Table 4:

Table 4 – Evaluation of the Candidate Computing Architectures Against the KPPs

Key Performance Parameters (KPPs) – Scoring	Rad-hard General Purpose Multicore	Rad-hard DSP Multicore	Rad-hard Reconfigurable Computing	COTS-based Multicore	Rad-hard Graphics Processing Units
Total KPP score (rank)	1	5	2	3	4
#KPP scores above mean	12/12	4/12	7/12	6/12	5/12

The metric on scoring above the mean was consistent with overall KPP scoring and particularly telling, and identified rad-hard general-purpose multicore as well above the other architecture candidates.

Advantages of a Multicore Architecture for Flight Computing

All of the evaluted architectures offer significant advances in computational performance. But rad-hard general purpose multicore scored highest on the KPP for cross-cutting potential across NASA missions. The reasons are based in the general-purpose character of the multicore architecture, as well as overall system-level flexibility.

The multicore architecture is natively receptive to power scaling at core-level granularity. Simply put, cores can be powered on and off to consume power efficiently as the thread-load dictates. The critical gap to address here is power dissipation at the hardware level and this technical tall pole is a focus of the risk-retirement activity currently underway.

Furthermore, multicore architectures are natively receptive to thread-based fault tolerance approaches. The cores themselves offer natural strategies for fault containment regions, directly supporting objectives for fault detection, correction and isolation. An ability to segregate failed cores from the pool of available cores directly supports graceful degradation. Finally, multicore architectures naturally support software-based N-modular redundancy (NMR) voting schemes for fail-operational fault tolerance. The underlying mechanisms for some of the required fault tolerance improvements are synergistic with or identical to the required power management modifications.

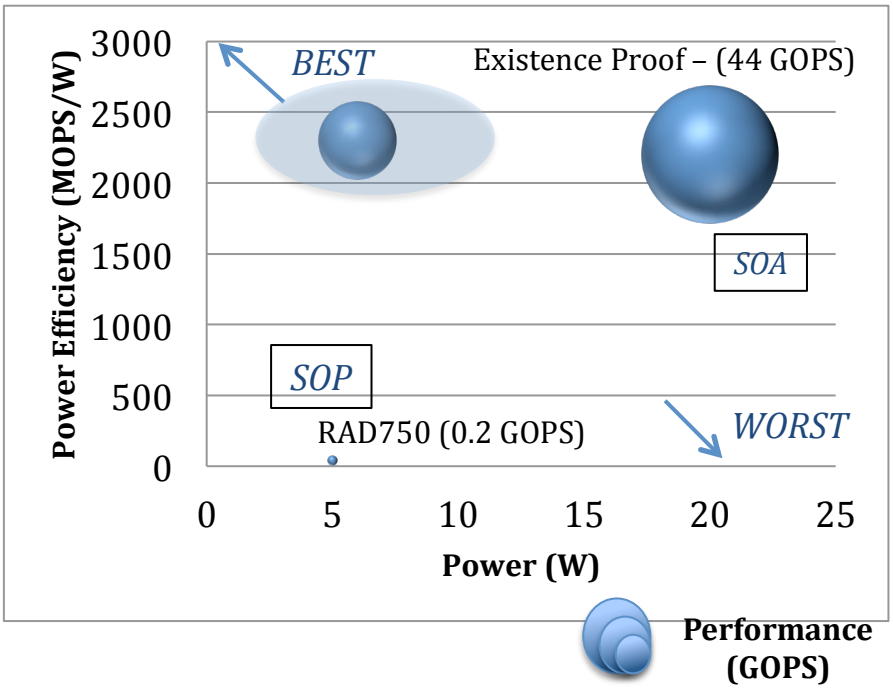
Multicore architectures are also inherently evolvable. As the technology for individual cores advances, a multicore flight computing system can be upgraded while retaining the overall architectural advantages.

Finally, a general-purpose multicore architecture may appear straightforwardly in a hybrid avionics systems architecture which might include reconfigurable FPGAs or specialized DSP/SIMD co-processor elements. Such a hybrid approach may be useful for specific missions where there is a preponderance of the type of processing at which these other architectures are aimed.

For all of these reasons: a significant advance in throughput, as well as efficiency and scalability, combined with the flexibility to dynamically trade system-level objectives for computational performance, energy management and fault tolerance, make rad-hard general-purpose multicore truly a game changing opportunity for effectively redefining flight computing for space systems.

Figure 1 depicts how a multicore flight computing solution can dramatically improve both computational performance and computational efficiency over the state-of-the-art (RAD750), as well as support dynamic choice of operating point – trading computational performance with energy management objectives (and fault tolerance - not shown) throughout the mission – a type of architectural support and flexbility that has never been available in NASA flight systems.

Figure 1 – Multicore Capability Advance and Support for Dynamic Operation



NASA-AFRL Partnership

NASA made the decision in December 2012 to proceed into an implementation phase for the HPSC project. A concurrent factor in devising an investment plan was a series of discussions with AFRL/Kirtland (Albuquerque, NM) that revealed similar interests and objectives concerning future flight computing capability, and a first-order alignment of requirements. As part of the same decision meeting, AFRL and NASA entered into partnership and issued a joint BAA in April 2013 entitled Next-Generation Space Processor (NGSP) Analysis Program. This BAA solicited, in an initial study phase focusing on technical risk retirement, architectural designs for a rad-hard general-purpose multicore flight computer [8,9] that address the derived requirements from the NASA study and additional needs to be determined by AFRL.

Three contractors were selected in September 2013 under this joint BAA: BAE, Boeing and Honeywell. The Kick-Off Meeting for the so-called Innovation Phase, to be one year in duration, was held in November 2013.

Innovation Phase

Part A of the Innovation Phase involved identifying US Air Force (USAF) use cases and derived requirements, to match the similar work previously performed by NASA. The vendors reported on the identified USAF use cases and derived requirements at the Technical Interchange Meeting held at NASA GSFC in February 2014. The use cases confirmed the relevance for a next-generation flight computing capabiltiy for the USAF. The reported requirements refined or augmented the existing set of requirements from the BAA. NASA and AFRL undertook an exercise to re-integrate flight computing requirements from the two agencies, which completed in March 2014. These updated requirements have been passed back to the vendors.

Part B of the Innovation Phase is the principal portion of the effort. The vendors are developing hardware architecture designs from the set of requirements and will evaluate those designs (primarily using models and simulations, given the short period of performance) against a set of NASA-provided benchmarks. The benchmarks are meant to provide a complementary means, beyond written requirements, to evalute the efficacy of the hardware architecture designs, and to expose any lurking issues – issues that have been known to slip through the letter of a requirements set, notwithstanding due diligence applied by both customer and vendor alike. The benchmarks in turn are derived from the identified NASA use cases. At the time of this writing, AFRL is embarking on their own effort to develop and provide benchmarks. The objective in using benchmarks is to improve the overall robustness of the hardware architecture design – consistent with executing an explicit risk retirement phase within the investment.

Benchmarks

The majority of benchmarks used in the industry today are best characterized as "kernel benchmarks." These typically consist of a single algorithmic "kernel" such as an FFT that is supplied as a short section of code. The intent of these benchmarks is to evaluate the performance of the processing system when executing this type of algorithm. In some cases, multiple kernels are ganged together in a suite to provide a more comprehensive evaluation of the processing system over a broader range of algorithms or usage parameters, usually related to a specific type of processing or application area.

In high performance computing (HPC), for example, the HPL Linpack TPP benchmark is used to measure the throughput of a machine running a floating point execution of a set of linear differential equations while the DGEMM benchmark measures the floating point execution rate of double precision real matrix-matrix multiplies. The STREAM benchmark measures memory bandwidth (usually the limiting factor) in a simple vector kernel, and PTRANS, a parallel matrix transpose, is used to evaluate the communication capacity of the network stitching together

multiple processors or processor cores in a multi-processing architecture. These plus a few other benchmarks are bundled into the High Performance Computing Challenge benchmark suite which is intended to provide a measure of overall machine performance for typical applications in high performance computing.

While these benchmarks are useful in evaluating a computing architecture, they can miss essential interactions and dependencies encountered in real world embedded applications. For this reason, our team decided to develop a set of benchmarks to represent complete applications rather than just kernel algorithms. By using complete realistic high performance applications, we intend to expose limitations that would not show up in highly focused kernel benchmarks. For example, many parallel applications nonetheless have elements that cannot be parallelized and must be run in serial fashion, thus creating a potential computing bottleneck. Many signal processing applications have general purpose computing sections that specialized signal processing architectures do not handle well, resulting in inefficiencies. Many future applications in embedded high performance computing are expected to be heavily database- and search-oriented, a computational paradigm that is not well represented in the typical HPC benchmark suite and that can engender communication bottlenecks in embedded processing architectures.

Most currently available benchmarks are evaluated by: 1) executing them in the machine being studied, and 2) measuring execution time. In parallel processing benchmarks, a scaling measurement may be needed to understand the scalability of the application in the architecture and to characterize the eventual computational bottleneck. In embedded computing applications, a power measurement may be thrown into the mix to characterize the power impact of the scaling. However, issues such as the effects of fault tolerance on power, performance, scalability, overhead, and architecture optimization are almost never included in traditional computing benchmarks. Similarly, the effects of dynamic resource and power management with respect to overhead, delays, and power-performance scaling is not typically evaluated. Finally, most traditional benchmarks have a single "stress knob", i.e., a parameter that is modified to increase system loading, in order to determine when the system slows or fails. Typically, the metric used is the number of computations being performed or a measure the size of the computation being performed. These alone do not accurately represent real world conditions in which a multiplicity of different parameters may stress the system in different ways, and which may engender different stress responses from the system. Examples include different types of fault conditions, increases in serial operations, and reduced real time response deadlines.

Each benchmark in the HPSC benchmark suite is representative of a class of applications that is expected to become more prevalent in future embedded computing systems in air, space, near-space, and surface (or subsurface) based platforms. Each benchmark represents a complete application: initialization, execution and termination clean up. Each benchmark is supplied with an input data set, a reference implementation in the form of C or C++ code, and an output file. The

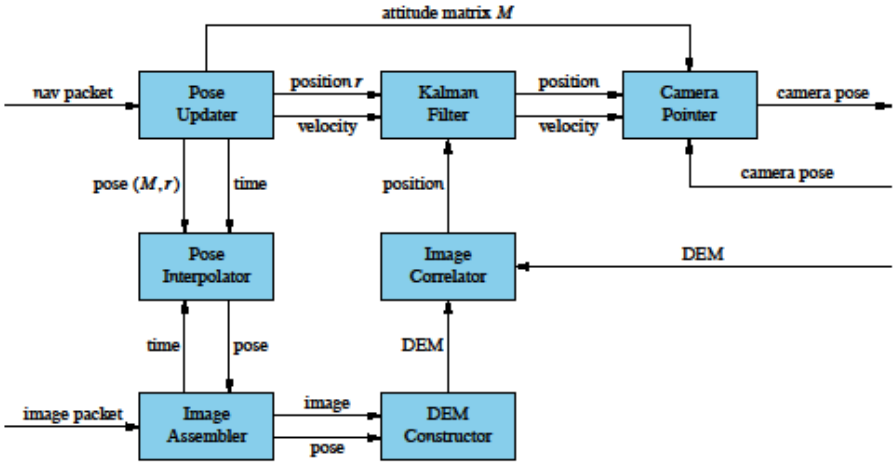
core of the benchmark is a specification. It is important to note that the reference code is just that, a reference, and is not meant to be run as-is on the target machine. Rather, the reference code is supplied to ensure clarity. The intent is that the benchmark be coded by the performing organization in a manner that is optimal for the machine being evaluated. The benchmark comprises: 1) an overall specification indicating the functionality of the benchmark, its overall structure, conditions under which it is to be executed and measurements to be made, and 2) a set of component specifications. A benchmark is thus built from a set of "components", each of which is a specification of one functional element of the overall benchmark. Each component specification consists of an introduction describing its functionality, and a detailed explanation of the intended operation, including flow charts and/or pseudo-code. Each component specification also contains an input data set and output data set as well as reference code. Wherever possible, generic codes, available from off-the-shelf libraries, are substituted for actual algorithms that might have been used in the application upon which the benchmark is based in order to avoid the publication of proprietary or ITAR restricted algorithms. We are in the process of building up a library of these components from which any application benchmark may be built. Many of these components are, in fact, built upon existing kernel benchmarks.

The HPSC benchmarks developed to date include:

Synthetic Application Benchmark 1

Models elements found in applications for vision-based real-time guidance, navigation and control of autonomous vehicles. This benchmark comprises seven components and is provided along with reference code, input data set and an output data set. Stress knobs include: application of extraneous additional computational load that must execute in parallel with the benchmark, image size, fault tolerance modes, and data rates. A block diagram is shown in Figure 2.

Figure 2 – Benchmark Diagram



Synthetic Application Benchmark 2

Models elements found in artificial intelligence applications such as autonomous planning and scheduling. This benchmark comprises seven components, each of which is based on a unique search algorithm. Input, output data sets and reference codes are provided.

Application Benchmark 2A: Automated Scheduling and Planning Environment

This complete application is provided as part of Synthetic Application Benchmark 2. It comprises an optimized application code, rather than reference code, which may be run as-is on a single processor along with specific input, output and test scripts, each script corresponding to a unique planning or scheduling scenario. Multiple instances of this benchmark may be run concurrently on a multiprocessor.

Synthetic Application Benchmark 3

Models elements found in synthetic aperture radar (SAR) type applications. This benchmark is supplied with input, output data sets and links to reference codes.

Synthetic Application Benchmark 4

This benchmark is currently under study and is planned for release later in 2014. This benchmark will contain elements found in autonomous vision-based science applications including signal processing, image processing, pattern recognition, and database search.

Functional Benchmark 1

This benchmark comprises a set of specific usage conditions and tests targeted at exposing the operational limitations of machine architecture characteristics such as: power scalability, communication network delays and saturation effects, interrupt distribution delays, fault tolerance mechanisms, and memory access times and bandwidth. Each element in this benchmark consists of a machine configuration setup, an execution specification and a measurement suite. At this time, reference codes are not planned as each machine will be significantly different and the benefit of reference codes is not clear.

Development Phase

The goal of the Innovation Phase is to confirm that at least one vendor has provided a hardware architecture design shown to responsive to the joint set of AFRL/NASA requirements, and furthermore that sufficient technical risk has been retired within that design. At the conclusion of the Innovation Phase, a review will be conducted to execute a Go/No-Go decision as to whether further investment in a next-generation

flight computing system is warranted. Assuming a positive outcome, AFRL and NASA plan to continue their partnership into a Development Phase.

The Development Phase is expected to run through three years, from approximately the middle of 2015 to the middle of 2018. The targeted product from this final phase of the investment is a board-level realization of a rad-hard general-purpose multicore flight computing system, with the following features:

- 100X computational performance of the RAD750
- <7W power budget, scalable
- Support for a range of fault tolerance methods
- Interoperable with co-processors

The board-level product will include the multicore chip, bundled with a real-time operating system (RTOS), hypervisor, and flight software (FSW) development environment (C/C++, debugger). In addition, NASA will develop a layer of middleware to be integrated with the hardware / system software stack as provided by the vendor.

The purpose of the middleware is to provide the capability to manage the cores dynamically for varying purposes, and to optimize the exploitation of a multicore architecture for the future NASA applications, as captured by the set of benchmarks. Specifically, the middleware provides the means to allocate and manage processor cores to dynamically trade computational performance, energy management, and fault tolerance objectives of the mission. The middleware layer will embody a resource allocator, power manager and fault tolerance manager. This concept for middleware can be seen as the logical end-point of a progression that begins with the use cases, proceeds through the derived requirements, is expanded to greater fidelity via the benchmarks, and arrives finally at the middleware. The common theme is to emphasize throughout the investment lifecycle that a flight computing system is as much about software as it is about hardware; the middleware is an important component to ensure that the system-level realization of the investment is well tuned to hosting the future flight *software* applications which will enable new mission capability and scenarios.

The investment product that a space mission will see is the board-level integrated hardware / system software / middleware stack that comprises the next-generation multicore flight computing system, along with a reference design kit. At this point, the HPSC capability will be available for validation and first mission use by NASA and the USAF.

Summary

We have reported here on status and plans of the joint investment between NASA and AFRL to develop a next-generation flight computing system to address the future space mission needs of the two agencies. NASA and AFRL intend to continue in partnership through a final development phase, and are seeking additional programmatic partners for this national-level capability investment.

It has been more than 15 years since NASA has invested in a flight computer. Our effort with AFRL provides a clear and broad basis for concluding that a next-generation flight computing capability will be a necessary and timely ingredient for future mission success. We are focusing on a multicore solution because of architectural support for dynamic operation, in addition to a signfiicant computational performance advance. Partnered with AFRL, we are taking large steps toward realizing a technology investment that will be truly game changing.

Acknowledgement

This research was carried out in part at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (NASA), funded by the Game Changing Development Program of the NASA Space Technology Mission Directorate.

References

1. Raphael Some, Richard Doyle, Larry Bergman, William Whitaker, Wesley Powell, Michael Johnson, Montgomery Goforth, and Michael Lowry, "Human and Robotic Mission Use Cases for High-Performance Spaceflight Computing," *AIAA Infotech*, Boston, MA, August 2013.
2. David R. Thompson, Daniel Q. Tran, David McLaren, Steve A. Chien, Larry Bergman, Rebecca Castano, Richard Doyle, Tara Estlin and Matthew Lenda, "Autonomous Onboard Science Data Analysis for Comet Missions," *11th International Symposium on Artificial Intelligence and Robotics Applications for Space,* Torino, Italy, September 2012.
3. James Alexander, Bradley J. Clement, Kim P. Gostelow, John Y. Lai, "Fault Mitigation Schemes for Future Spaceflight Multicore Processors, *AIAA Infotech*, Garden Grove, CA, June 2012.
4. James Alexander, Yang Cheng, William Zheng, Nikolas Trawny and Andrew Johnson, "A Terrain Relative Navigation Sensor Enabled by Multi-Core Processing," *IEEE Aerospace*, Big Sky, MT, March 2012.
5. Peter M. Kogge, Benjamin J. Bornstein and Tara A. Estlin, "Energy Usage in an Embedded Space Vision Application on a Tiled Architecture," *AIAA Infotech*, St. Louis, MO, March 2011.
6. Kim P. Gostelow, "The Design of a Fault-Tolerant, Real-Time, Multi-Core Computer System," *AIAA Infotech*, St. Louis, MO, March 2011.
7. Wesley A. Powell, Michael A. Johnson, Jonathan Wilmot, Raphael Some, Kim P. Gostelow, Glenn Reeves and Richard J. Doyle, "Enabling Future Robotic Missions with Multicore Processors," *AIAA Infotech*, St. Louis, MO, March 2011.
8. Carlos Villalpando, Raphael Some, et al., "Reliable Multicore Processors for NASA Space Missions", *IEEE Aerospace*, Big Sky, MT, March 2011.
9. Carlos Villalpando, Raphael Some, et al., "Investigation of the Tilera Processor for Real Time Hazard Detection and Avoidance on the Altair Lunar Lander", *IEEE Aerospace*, Big Sky, MT, March 2010.
10. Richard Doyle, Raphael Some, Wesley Powell, Montgomery Goforth, David Guibeau and Michael Lowry, "High Performance Spaceflight Computing, An Avionics Formulation Task, Study Report Executive Summary, October 2012.