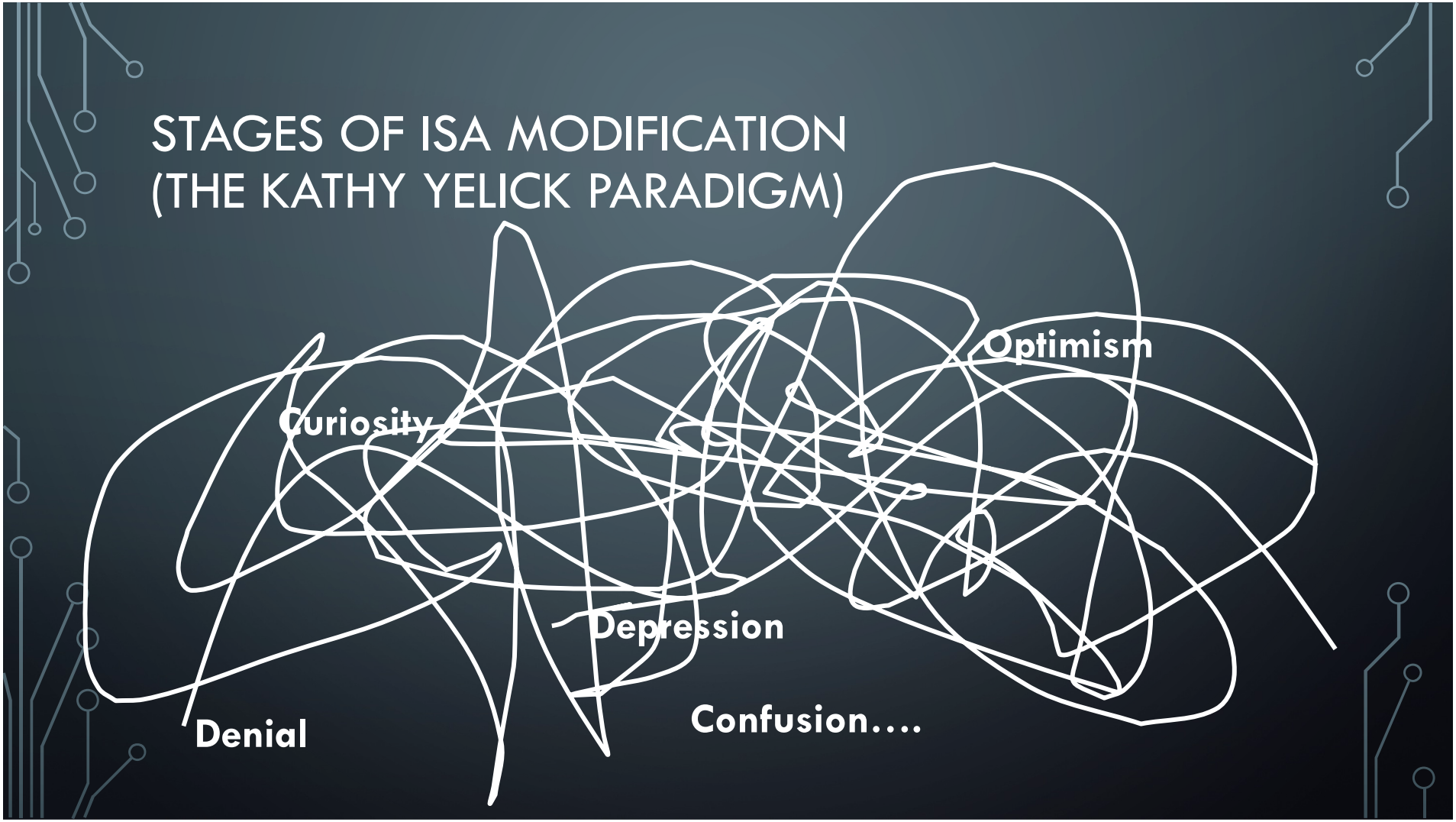
IF YOU COULD HAVE ANY INSTRUCTION, WHAT WOULD IT BE ?

JOHN LEIDEL

CHIEF SCIENTIST, TACTICAL COMPUTING LABS

SALISHAN 2024

STAGES OF ISA MODIFICATION (THE KATHY YELICK PARADIGM)



WHAT IS AN ISA?

An Instruction Set Architecture (ISA) is part of the abstract model of a computer that defines how the CPU is controlled by the software. The ISA acts as an interface between the hardware and the software, specifying both what the processor is capable of doing as well as how it gets done.

https://www.arm.com/glossary/isa

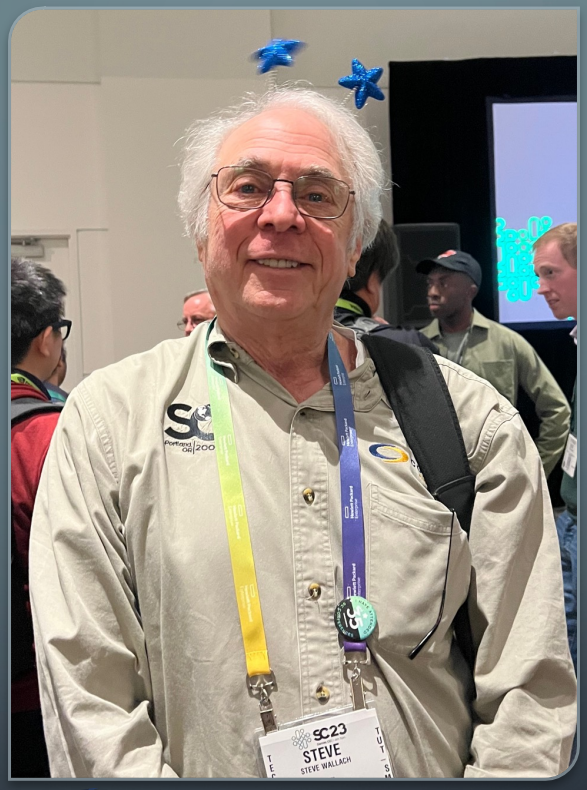
WHAT IS AN ISA?

An Instruction Set Architecture (ISA) is part of the abstract model of a computer that defines how the CPU is controlled by the software. The ISA acts as an interface between the hardware and the software, ~~specifying both what the processor is capable of doing as well as how it gets done~~.

https://www.arm.com/glossary/isa

HOW DID I GET HERE?


KNIGHT



CONVEY COMPUTER™

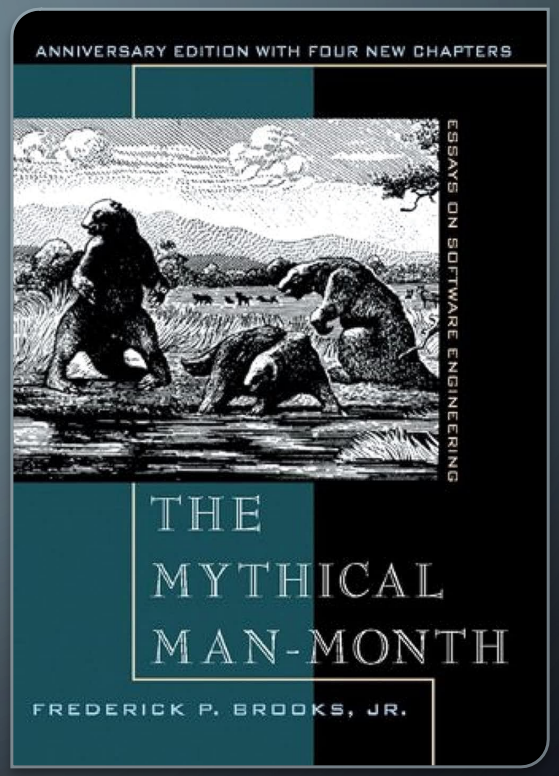
A LITTLE ISA HISTORY…

HERE BE DRAGONS

https://education.nationalgeographic.org/resource/here-be-dragons/

HISTORY OF ISA'S

- Fred Brooks & the IBM System/360
  - Backward compatibility of execution between models
- *"The SPREAD compatibility objective, in contrast, postulated a single architecture for a series of five processors spanning a wide range of cost and performance."*



THINKING MACHINES CM-5

- Arguably one of the first leadership class systems to integrate a commercial microprocessor and an accelerator
- SPARC + DPEAC (MIMD w/ simulated SIMD)
- The DPEAC ISA supported parallel communication operations in the form of unidirectional sends

1.4 A Simple C* Routine

The C* routine below performs both ordinary scalar C operations and parallel operations:

```
#include <stdio.h>
void fishcake(int x, int:current a, float:current b)
{
    float sum;
    x = x + 2;
    printf("The value of x is: %d\n", x);
    b = b * 17.2f + a * x;
    sum = += b;
    printf("The sum of b is: %f\n", sum);
}
```

5.1 Tips for Increasing General Communication Performance

5.1.1 Use Send Operations instead of Gets

If possible, write your program to use a send operation — for example:

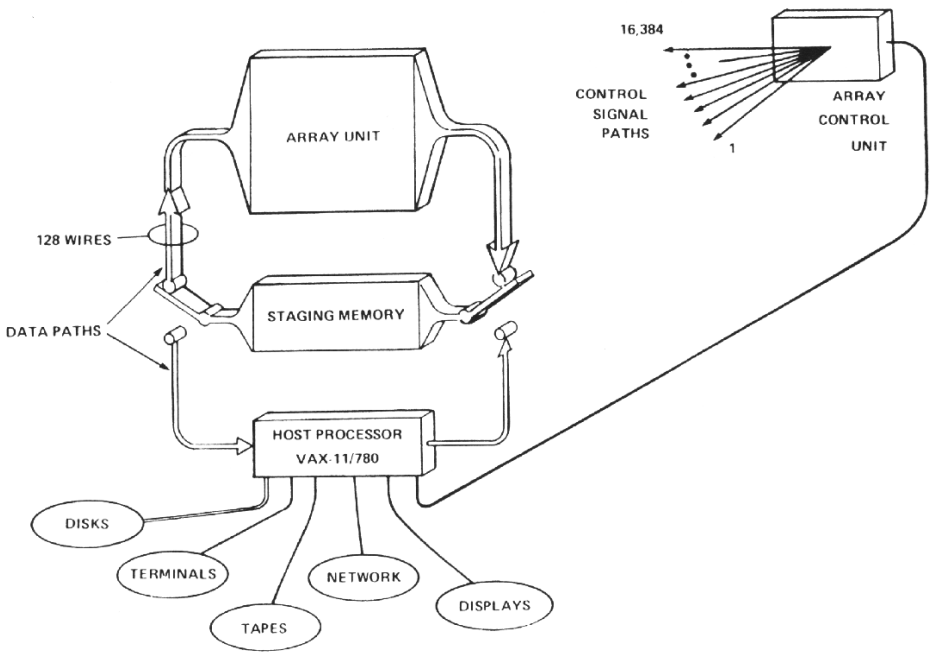
```
int:current index, dest, source;
[index]dest = source;
```

instead of a get operation — for example:

```
dest = [index]source;
```

The CM-5 hardware does not directly implement get operations; instead, the run-time system performs a send for the request and a send for the reply. Therefore, a get operation is roughly twice as expensive as a send.

https://dl.acm.org/doi/pdf/10.1145/1500175.1500260



BATCHER CORNER TURN

- Goodyear STARAN/Goodyear MPP systems were 1-bit processors configured in 2D arrays (128x128)
- Memory could be accessed as rows or columns with an internal "corner turn" engine that could transpose data in constant time

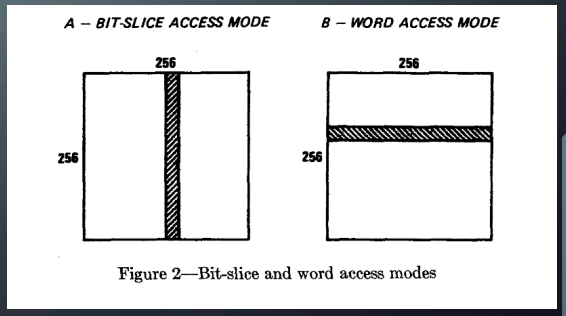
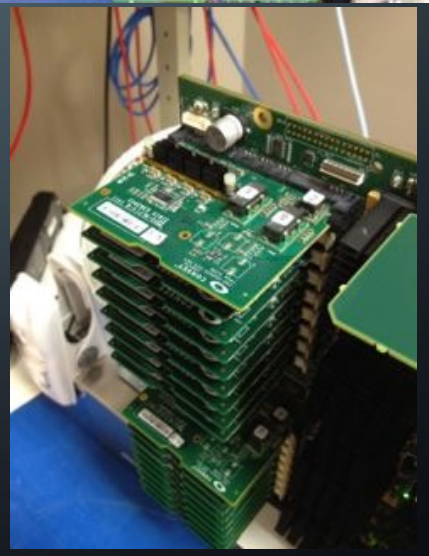


Figure 2—Bit-slice and word access modes

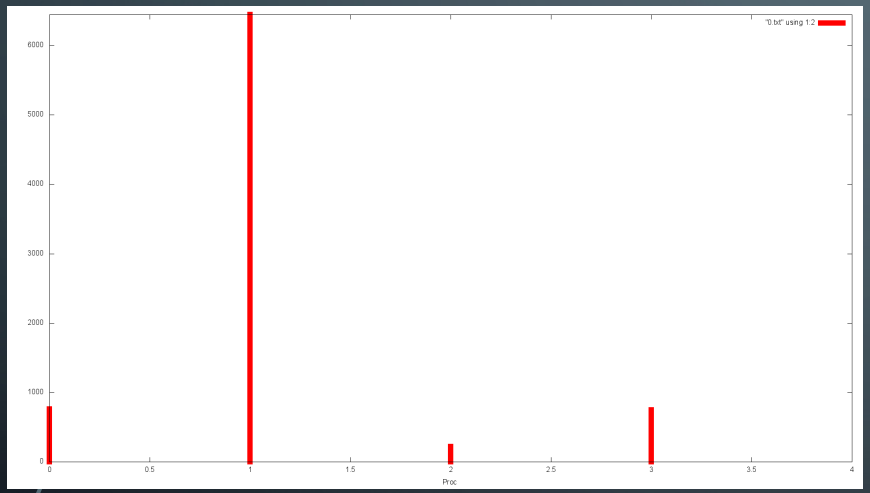
T. Finkbeiner, G. Hush, T. Larsen, P. Lea, J. Leidel and T. Manning, "In-Memory Intelligence," in *IEEE Micro*, vol. 37, no. 4, pp. 30-38, 2017, doi: 10.1109/MM.2017.3211117. keywords: {Random access memory;Computer architecture;VLIW;Vectors;Moore's Law;Computational modeling;Process control;Microprocessors;Memory management;processor in memory;non-Von Neumann;computer architecture;SIMD;vector processing},

CONVEY MX-100

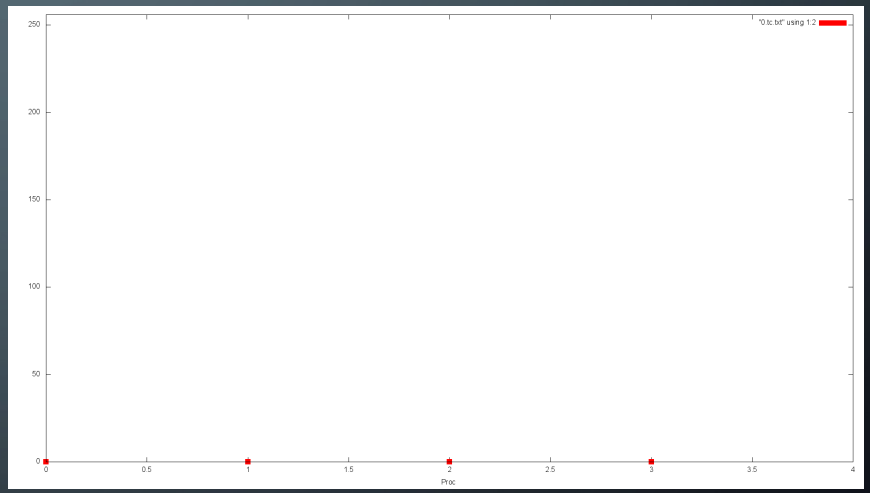
- Convey developed hybrid FPGA+Intel platforms
- FPGA's were loaded with "personalities" that encapsulated ISA's
- Convey historically developed long vector ISA's with extremely good vector ISA compiler support
- MX-100 platform was designed for sparse workloads (graphs, SPMV, etc)



CHOMP ISA



Traditional x86 thread load imbalance using work-stealing



MX-100 accelerated work-stealing thread oversubscription and balance

CHOMP ISA

- "Convey Hybrid OpenMP" utilized OpenMP threading/tasking as the primary driver for ISA development
- Where do we context switch!?
- Any single instruction could induce a thread context switch (work sharing) using a single bit in the instruction payload

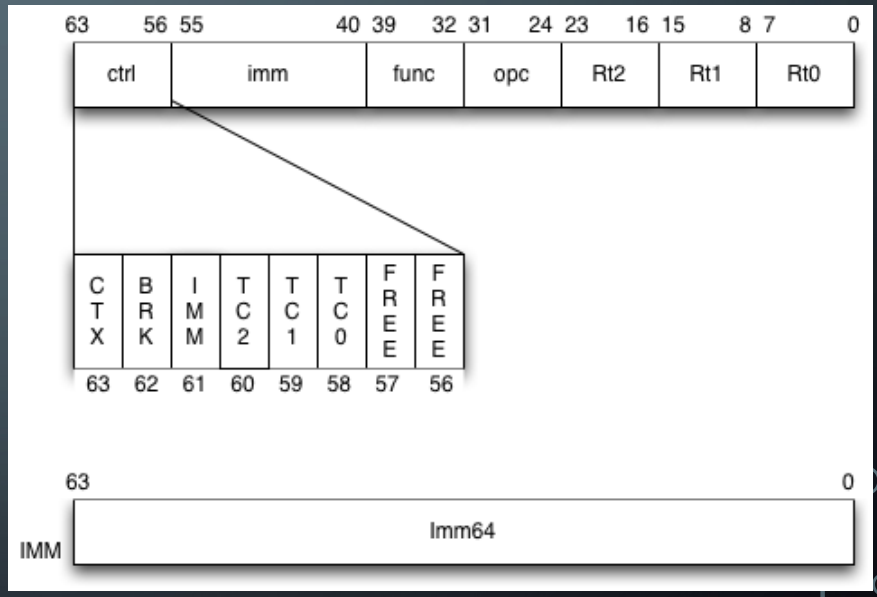
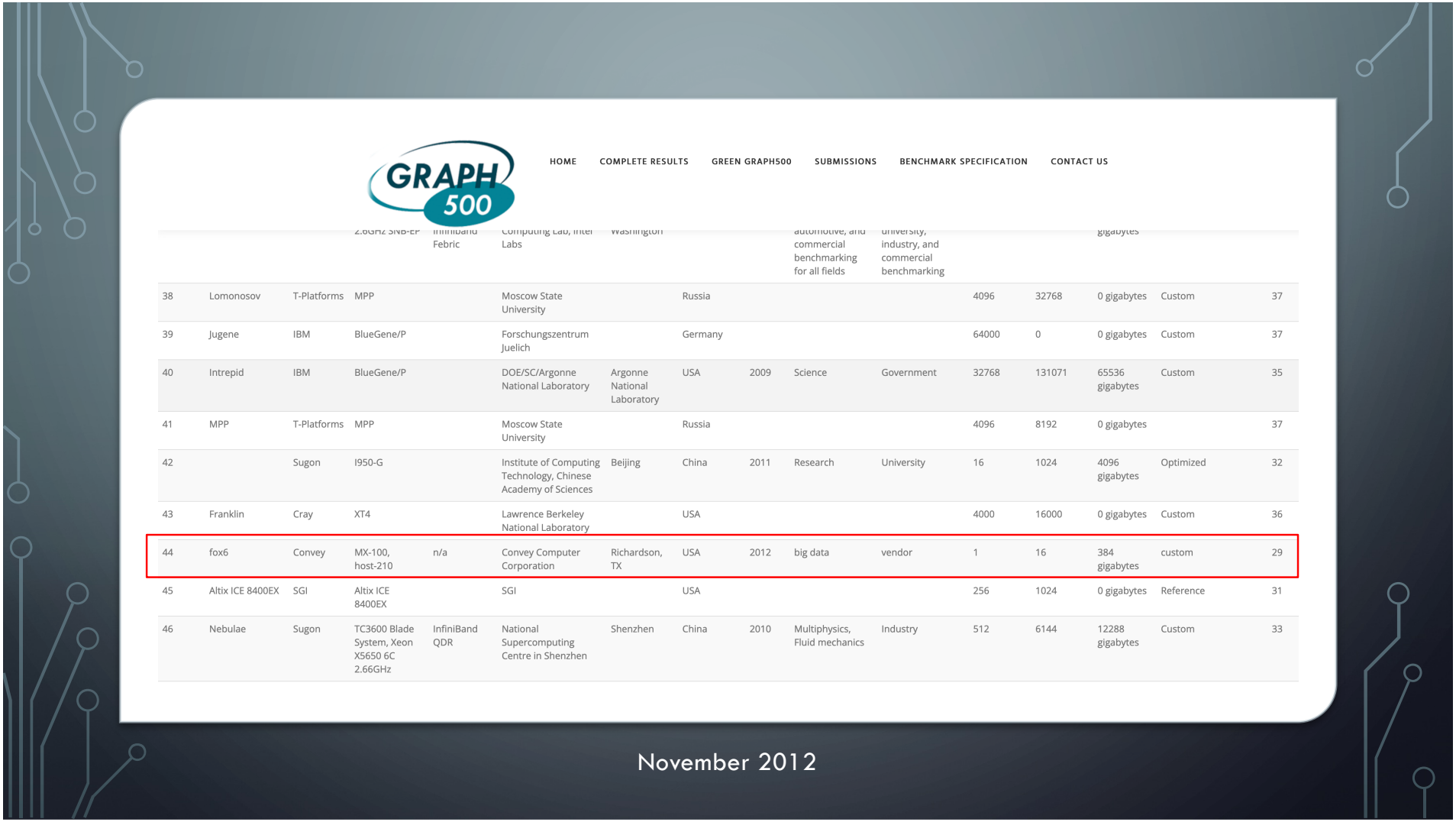


Figure N: CHOMP AEFP Instruction Format

CHOMP: A Framework and Instruction Set for Latency Tolerant, Massively Multithreaded Processors



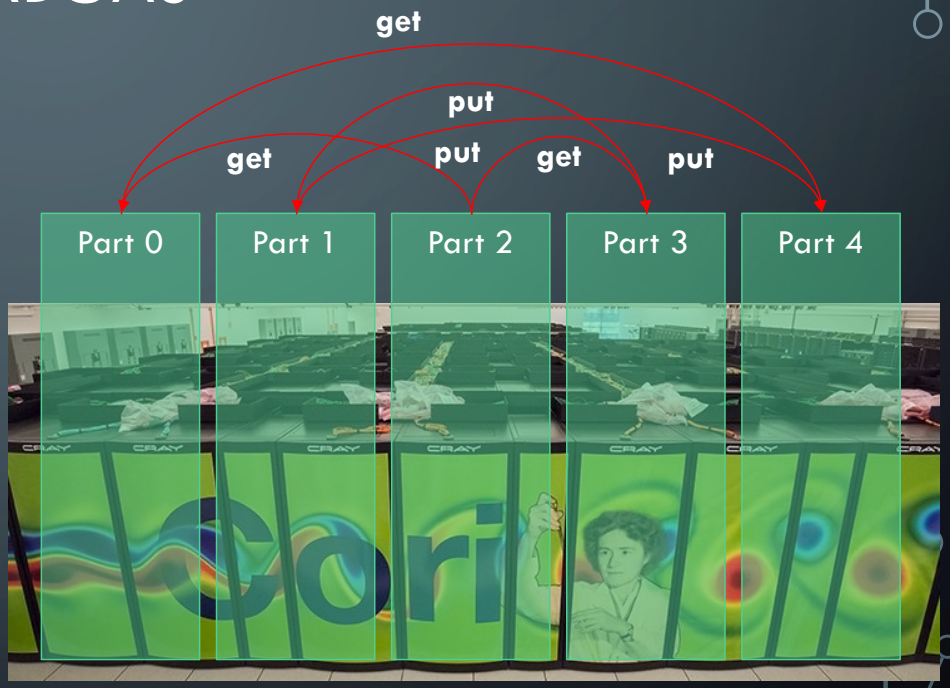
			2.6GHz SNB-EP	Infiniband Febric	Computing Lab, Intel Labs	Washington			automotive, and commercial benchmarking for all fields	university, industry, and commercial benchmarking			gigabytes		
38	Lomonosov	T-Platforms	MPP		Moscow State University		Russia				4096	32768	0 gigabytes	Custom	37
39	Jugene	IBM	BlueGene/P		Forschungszentrum Juelich		Germany				64000	0	0 gigabytes	Custom	37
40	Intrepid	IBM	BlueGene/P		DOE/SC/Argonne National Laboratory	Argonne National Laboratory	USA	2009	Science	Government	32768	131071	65536 gigabytes	Custom	35
41	MPP	T-Platforms	MPP		Moscow State University		Russia				4096	8192	0 gigabytes		37
42		Sugon	I950-G		Institute of Computing Technology, Chinese Academy of Sciences	Beijing	China	2011	Research	University	16	1024	4096 gigabytes	Optimized	32
43	Franklin	Cray	XT4		Lawrence Berkeley National Laboratory		USA				4000	16000	0 gigabytes	Custom	36
44	fox6	Convey	MX-100, host-210	n/a	Convey Computer Corporation	Richardson, TX	USA	2012	big data	vendor	1	16	384 gigabytes	custom	29
45	Altix ICE 8400EX	SGI	Altix ICE 8400EX		SGI		USA				256	1024	0 gigabytes	Reference	31
46	Nebulae	Sugon	TC3600 Blade System, Xeon X5650 6C 2.66GHz	InfiniBand QDR	National Supercomputing Centre in Shenzhen	Shenzhen	China	2010	Multiphysics, Fluid mechanics	Industry	512	6144	12288 gigabytes	Custom	33

November 2012

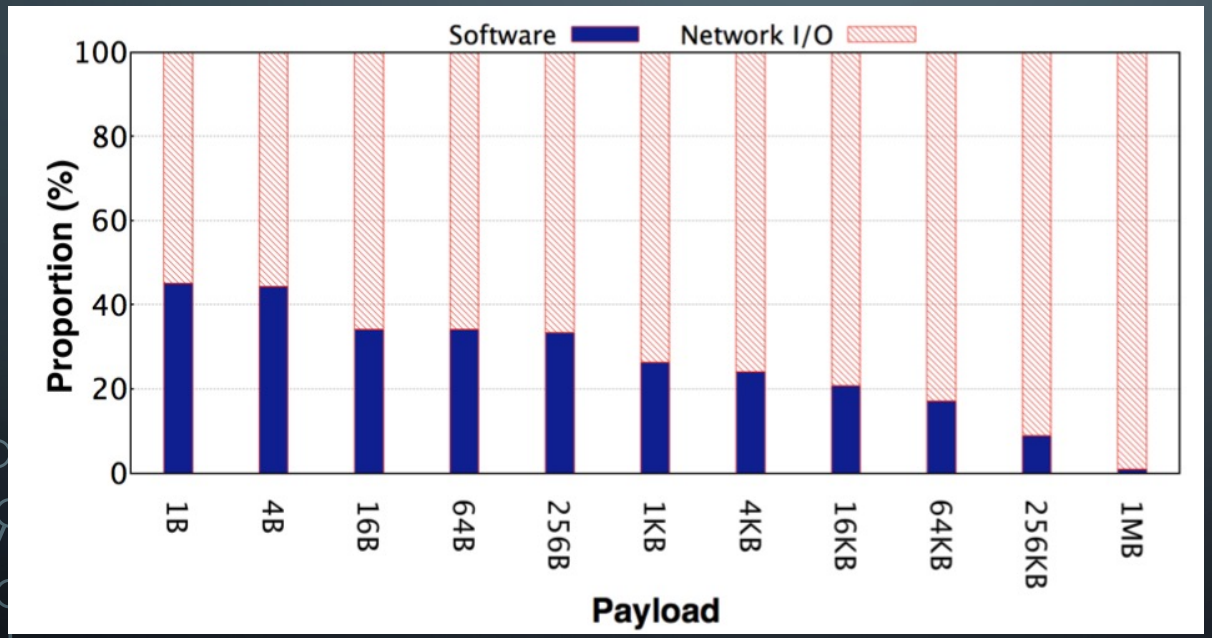
A LITTLE EXPERIMENT…

BACKGROUND BEHIND XBGAS

- Traditional message passing paradigm limitations
  - User library overhead, driver overhead
  - Optimized for large data transfers of regular workloads
  - Limited scalability for leadership-class systems
- Little hardware/uArch support in existing PGAS paradigms!
- TTU, TCL, ASU collaboration



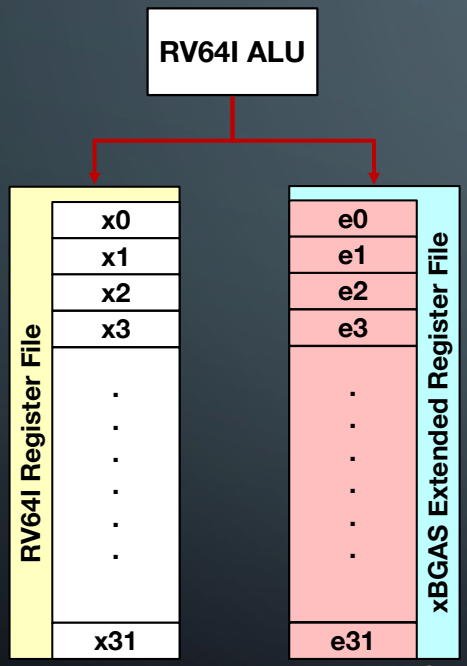
DATA MOVEMENT IS EXPENSIVE!



OpenSHMEM Get Profiling (OSHMEM 3.0.4 + UCX 1.6.0)

XBGAS: A global address space extension on RISC-V for high performance computing

XBGAS ARCHITECTURE



Extended xBGAS Registers e0 – e31

I-Type				
Mnemonic	Base	Funct3	Dest	Opcode
eaddie extd, rs1, imm	rs1	111	extd	1111011
eld rd, imm(rs1)	rs1+ext1	011	rd	1111011

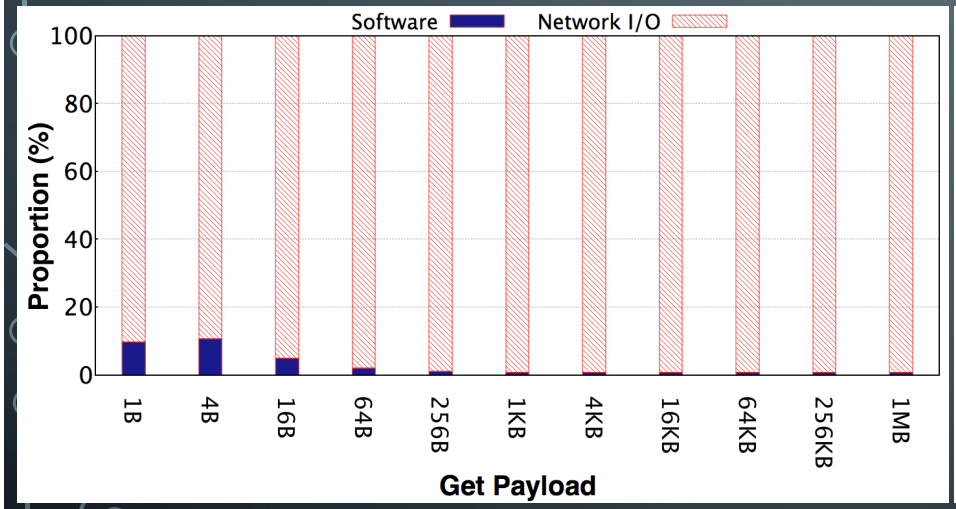
R-Type						
Mnemonic	Funct7	RS2	RS1	Funct3	RD	Opcode
erld rd, rs1, ext2	1010101	ext2	rs1	011	rd	0110011
ersd rs1, rs2, ext3	0100010	rs2	rs1	011	ext3	0110011

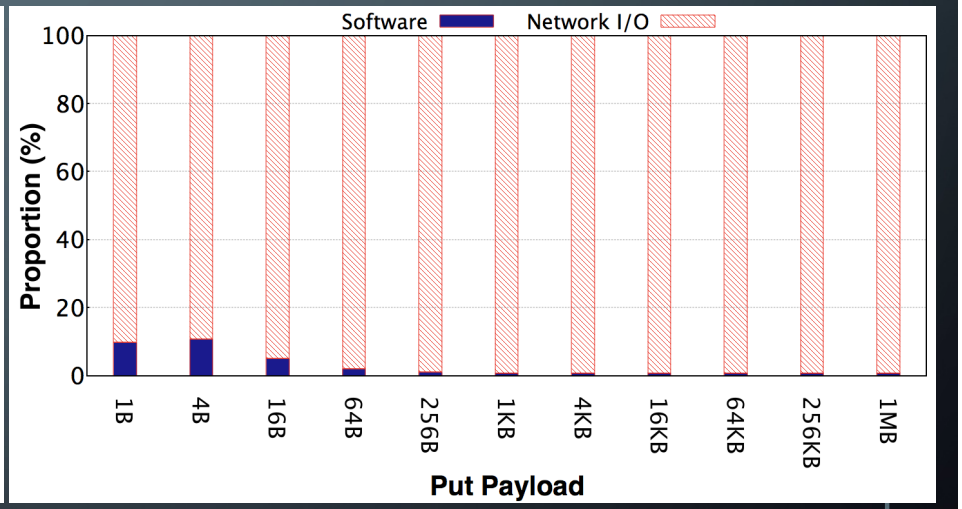
S-Type				
Mnemonic	Src	Base	Funct3	Opcode
esd rs2, imm(rs1)	rs2	rs1+ext1	011	1111011
esw rs2, imm(rs1)	rs2	rs1+ext1	010	1111011

xBGAS Instructions are split into

- Ext. address management
- Base integer load/store
- Raw integer load/store
- Remote atomic instructions

XBGAS PERFORMANCE

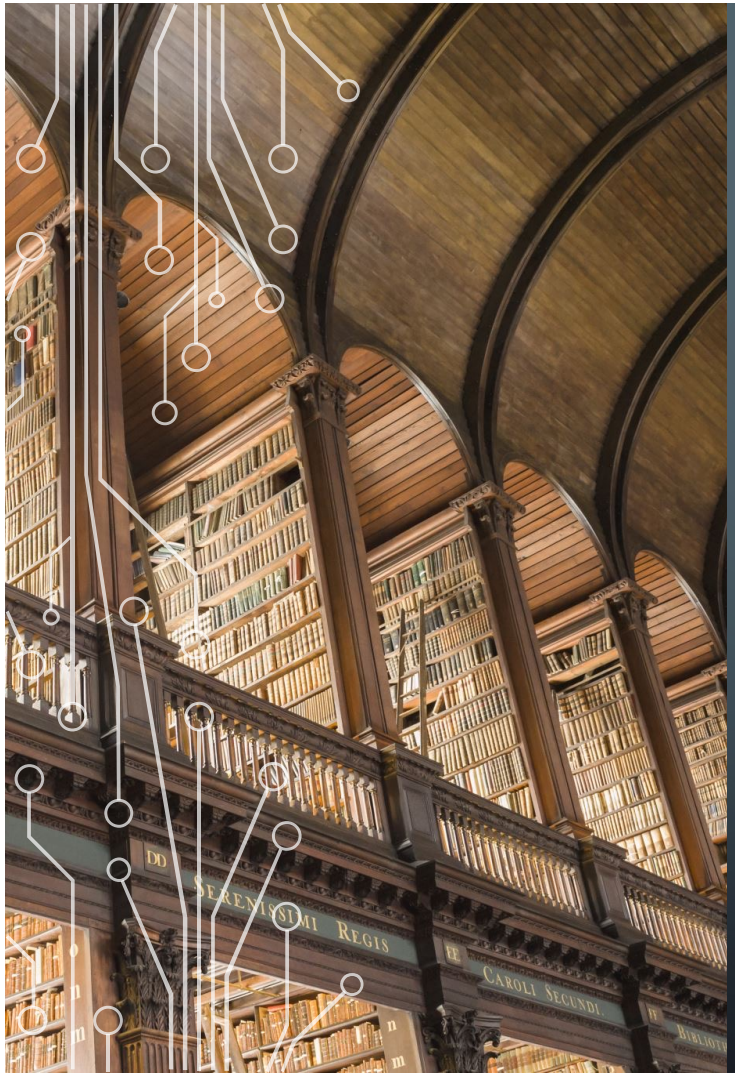




xBGAS significantly reduces the software cost of accessing a remote register-width data element to **9.7%** of that demonstrated by OpenSHMEM

Implies a software overhead reduction of **78.43%**

A CHALLENGE FOR THE FUTURE



HPC IN THE FUTURE: "BUSINESS-AS-USUAL WILL NOT BE ADEQUATE"

- The entire system should be considered
  - Hardware density is incredible, but the balance of efficiency is terribly skewed!
  - Optimizing single aspects of future architectures is intractable
  - The contract between hardware and software needs to be revisited
- Explore the dark corners of applications, compilers and runtime libraries!

- ***Challenge yourselves to understand the "where" and "why" in the entire system as it reflects upon the application and the user***

ACKNOWLEDGEMENTS

- David Donofrio
- Steve Wallach, Tony Brewer, Bruce Toal
- Dr. Peter Kogge
- Martin Deneroff