Shared Memory HPC Programming: Past, Present, and Future

Bill Carlson
IDA Center for Computing Sciences
June 13, 2015

Disclaimer

This presentation reflects the personal views of the author.

These views may or may not be held the author's employer or its sponsors.

Our Problem in 1993

How do we program this? And get good performance?

AC for the Cray T3D

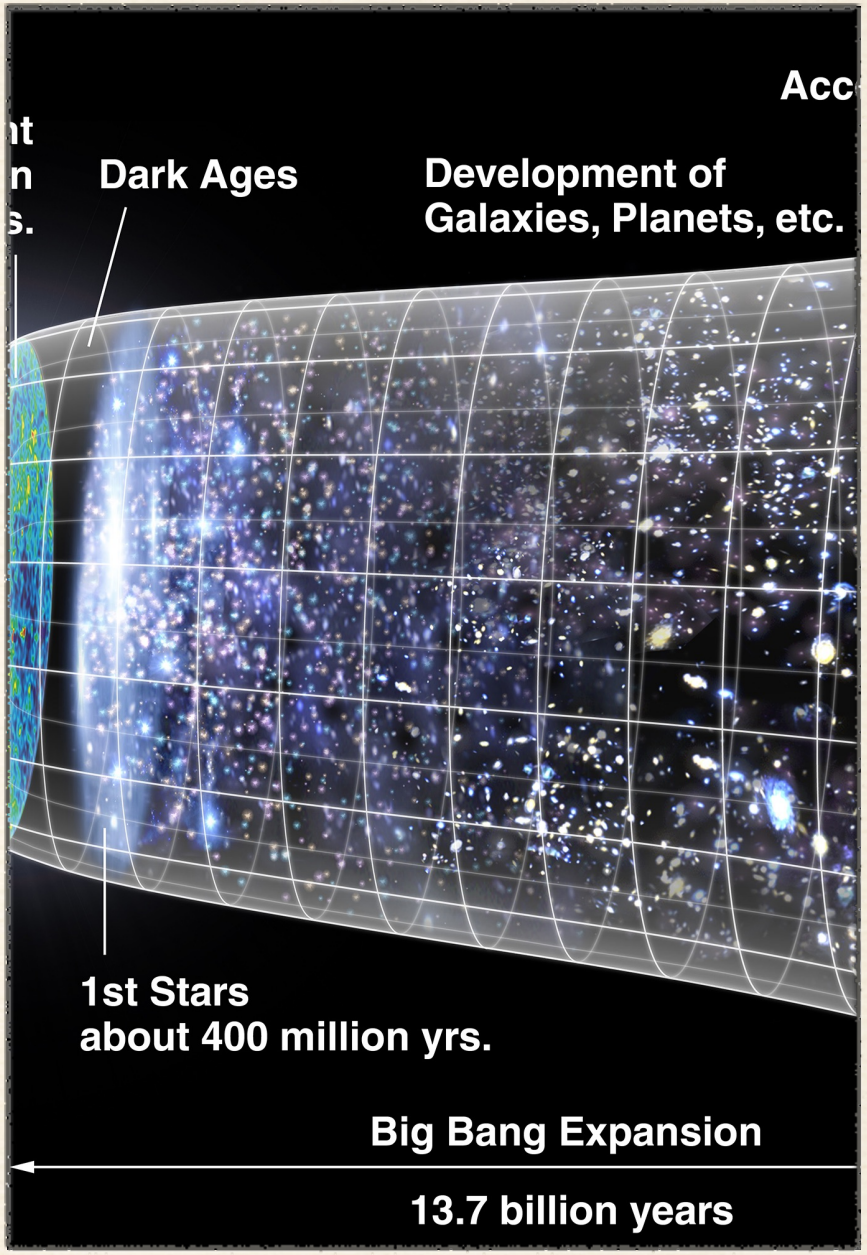
- An outgrowth of our work on CM5
- Shared memory on a distributed memory machine
  - "dist" keyword is the only syntax change
  - Performance high from special hardware on T3D
  - Much faster than "shmem" library, due to low overhead

UPC = AC + Split-C + PCP

- Collaboration with UC Berkeley and LLNL
  - Takes “shared” from AC’s “dist”
  - “strict” and “relaxed” shared memory semantics
  - Split barriers: “notify/wait”
  - Locks
  - Adds several data distributions

PGAS: Expanding the collaboration

- SHMEM Library
- CoArray Fortran
- Global Arrays
- Titanium



DARPA's HPCS Program

- High ***Productivity*** Computing Systems
- Productivity: Output per unit of Input
  - Output is problems solved
  - Input is money, energy, people time
- Goal: Increase productivity of HPC by 10x:
  - Systems performance 10x for many metrics
  - Algorithm and Software developers 10x effective in making good code
  - System operators spend 1/10 effort to manage system

HPCS added “modern” concepts

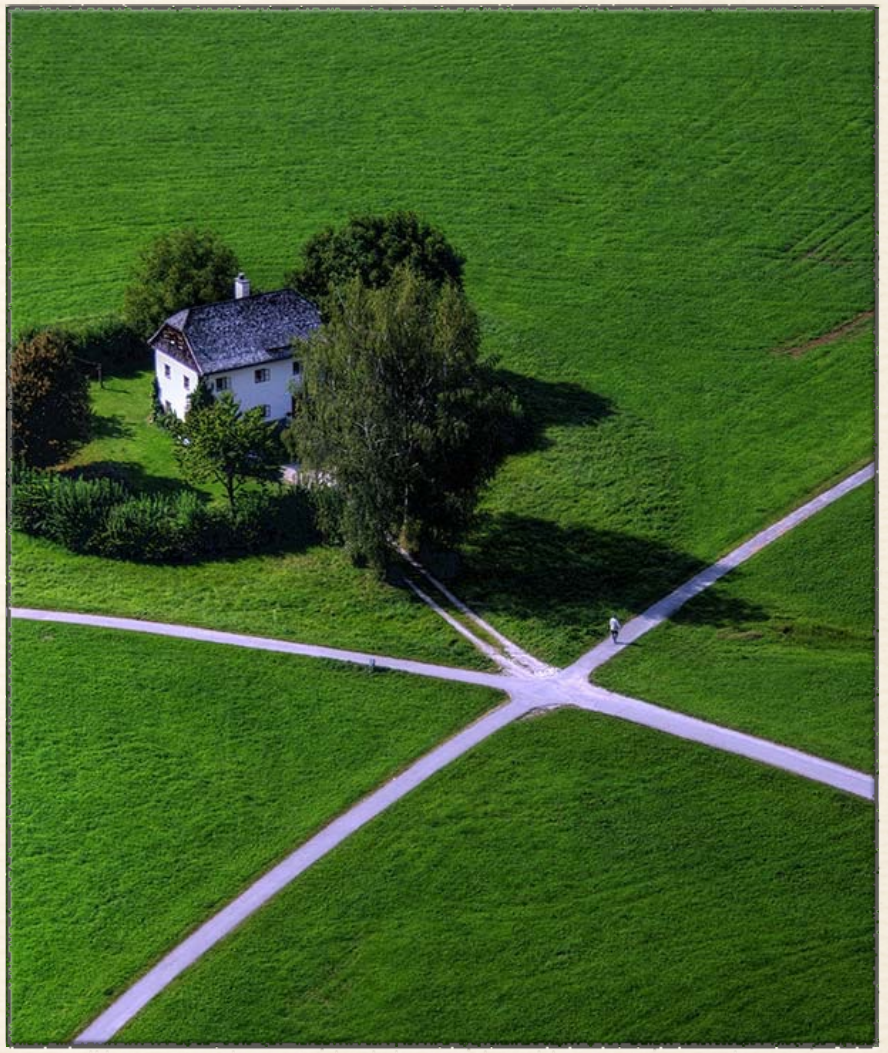
- Fortress: Implicit Parallelism, Strong Types
  - Cool look: like math in both ASCII and Unicode
  - Effort ended in 2012
- X10: Java-like syntax, asynchrony, locales
  - Going strong, a workshop at this conference
- Chapel: You all know this!

Post-HPCS PGAS efforts

- Habenaro C and UPC++ (Rice)
- UPC++ (Berkeley)
- CoArray C++ (Cray, EPCC)
- HPX (C++/11,14, LSU, FAU)
- XcalableMP (Tskuba)
- GASPI (Fraunhofer)

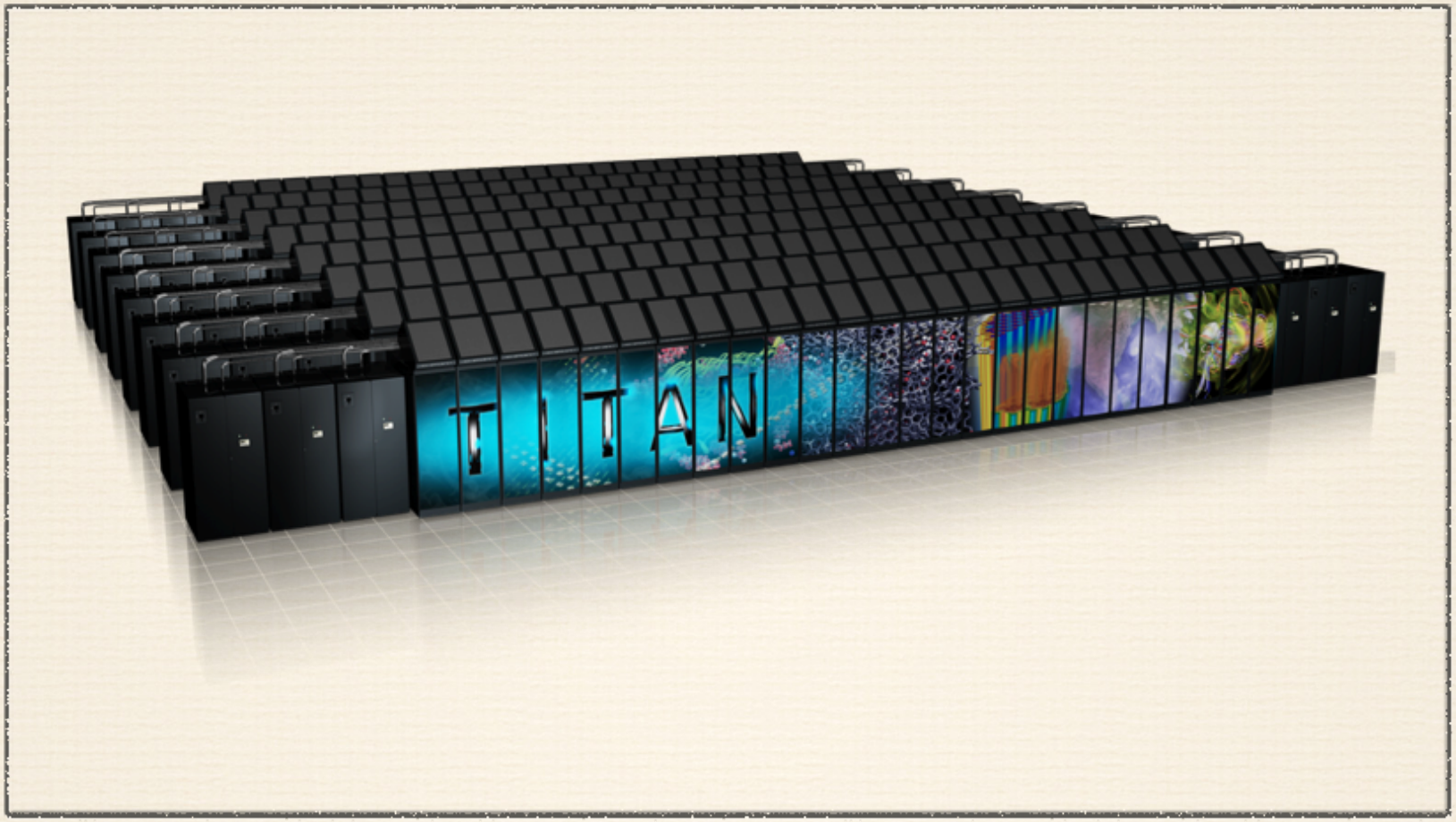
HPC at a Crossroads

- Path to ExaScale is underway
- Market is experiencing growth
- Systems increasingly specialized
  - Driven by ExaScale goals
- Application development is getting harder, not easier



PGAS at a Crossroads

- Many implementations exist of PGAS techniques
- Provide a wealth of programming metaphors
- Performance has been shown to be very good
  - A number of cases which exceed best message passing code
    - Because you have a wider choice of algorithms!
- Programmer “base” is
  - (somewhat) small, and
  - (somewhat) static



Our Problem in 2015

How do we program this? And get good performance? And expand use?

Thought Questions for Today

- Should the programming model be multi-level?
- Will future HPC systems be more complex?
- Can PGAS bring entire new use cases to HPC?
- Should PGAS care about HPC?

Multi-Level Parallelism?

- Hardware is becoming increasingly hierarchical
  - Start with SMP "nodes" in distributed machines
  - Add threads within cores within processors
    - GPUs and other accelerators only add to the mess
- Two distinct issues:
  - What is shared among threads on a "node"? But not globally?
  - What controls the parallel activity on a node?

Multi-Level Parallelism?

- Some programming models urge multi-level
  - SHMEM + pthreads or OpenMP
  - Programmers then write two levels of control flow, one for across nodes, one for on nodes
- UPC supports only local and shared
  - What is the problem with a PGAS thread per thread?
  - An extension was made to allow shared allocation on node

Complexity of Next Gen HPC?

- Strong forces for higher complexity
  - Need to control energy leads to specialization
    - Accelerators like GPUs
    - Small, specialized memories
    - Communication at a distance is always limited by cost
  - ExaScale goals are pushing for large performance gains
- Some trends to lower complexity
  - Many applications can fit “on a node” or small segment of system
  - Communication bound algorithms might ignore complex parts of system

New Uses for HPC/PGAS

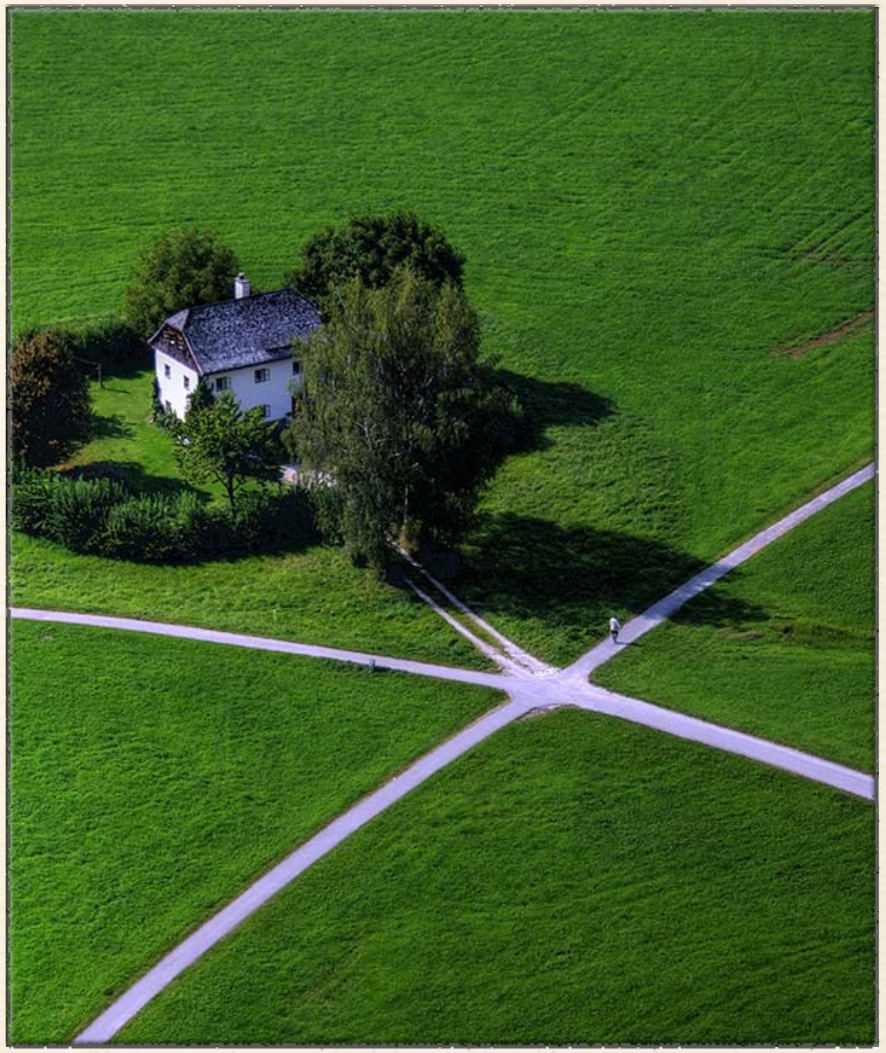
- A lot of emphasis on “Big Data”
  - How about an awesomely fast PGAS key-value store
- Machine “Deep” “learning”
  - Can PGAS allow real advances in this field
- Previously “Abandoned” HPC applications
  - Industrial uses in manufacturing
- My assertions:
  - PGAS languages could help add new application areas
  - All of these areas are not using HPC (much) because it is too hard to get apps on systems

Is “HPC” the only PGAS “market”?

- Mostly yes
  - Pointless to “partition” a tiny system
- But maybe not!
  - No widely-useful model for programing SMP processors
    - Most restricted to concurrency (e.g., go)
  - PGAS could provide a path to scalable apps
  - PGAS can be powerful metaphor in progammer education

PGAS Future?

- Stay the Course?
- Another Unification?
- Another Adaptation?



Path Forward One: Keep Pressing

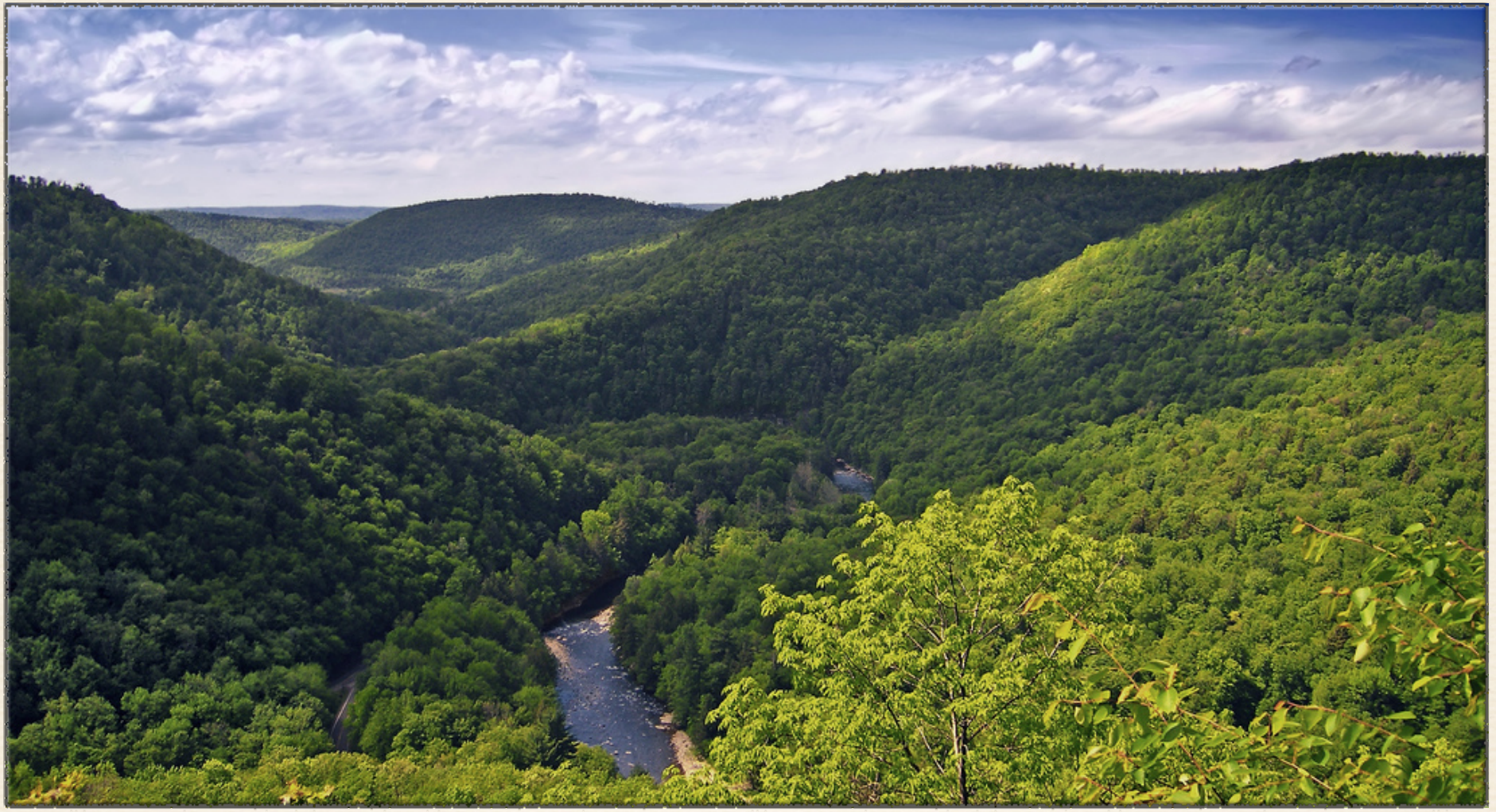
- Our current languages are good!
- Our current programmers are good!
- We are growing friends all the time
- To Do List:
  - Implement github-scale sharing of PGAS utilities
  - Start work on new application areas
  - Develop curriculum

Path Forward Two: New Unification

- UPC took three smaller, locally used languages
  - And made something better than sum of parts
- Many C++ based PGAS efforts are underway
  - And others have been considered as well
  - C++ recently gaining “popularity”
  - Recent changes in C++ standard help
    - Maybe admits a “PGAS class” without language change
- But gaining branding and adoption is always hard

Path Forward Three: New Adaptation

- Python
  - Very popular, including at many HPC centers
  - My view: current parallel classes poor fit for Python
    - Opportunity!
- Go (Google)
  - Already has a concurrency model, can parallel be added?
- Swift (Apple)
  - Will be a huge programmer base due to iPhone
- Any of these (and probably many others) could admit PGAS as a “class”



Future Vistas for PGAS

The fun has only begun

Image Credits

- T3D Image: CC-BY-SA-2.0-fr Rama, Wikimedia: CRAY-T3D IMG 8981-82-87-89.CR2.jpg
- Expanding Image: Public Domain, "CMB Timeline300 no WMAP" by NASA Wikimedia:CMB_Timeline300_no_WMAP.jpg
- Crossroads Image: CC-BY-SA Umberto Nicoletti, flickr.com/photos/unicoletti/2851575552
- Titan Image: Public Domain, James086, Wikimedia: Titan render.png
- Final Image: CC-BY-2.0 Nicholas A. Tonelli, flickr.com/photos/nicholas_t/6697202819