CRAY
THE SUPERCOMPUTER COMPANY

LLVM on 180k Cores

David Greene
Compiler Optimization & Codegen
Cray, Inc.

dag@cray.com

Outline

- Actually 181.504k Cores (but we like even numbers)

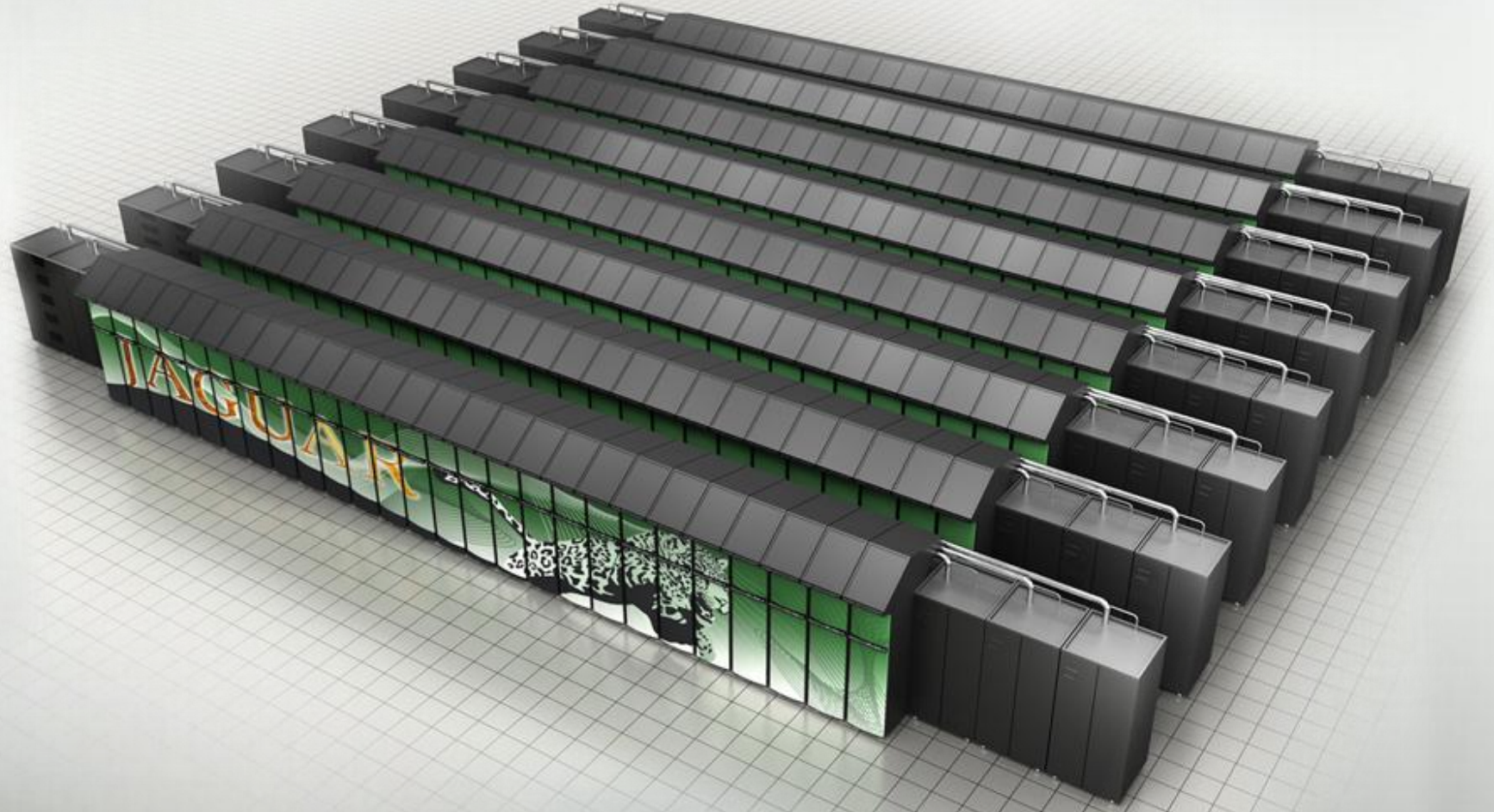
- What is High-Performance Computing?
- Challenges
- The Compiler's Role
- LLVM: What Works
- LLVM: What's Needed
- LLVM: Thinking Forward

What is HPC?

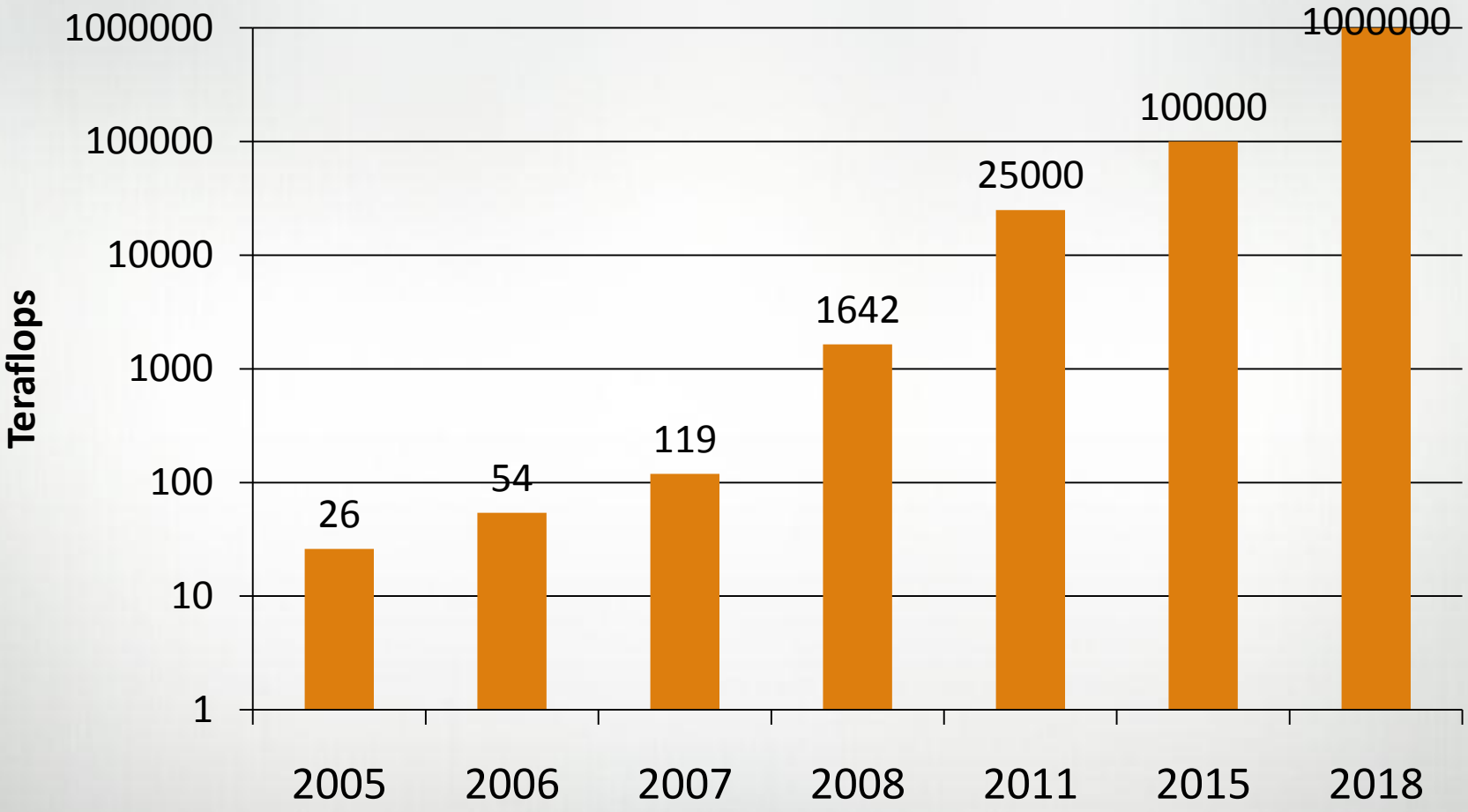
- Fast, balanced, scalable machines
  - Hundreds of thousands of cores
  - Petabytes of memory
  - Petaflops of processing power
- To push the envelope of science
  - Handling barely-solvable, otherwise intractable leading-edge problems
  - Customers that are willing to wrestle with us in the mud
- That are reasonably easy to use
  - Scientists should be scientists (not computer scientists)
  - Software is king

Case Study: Jaguar

- Installed at Oak Ridge National Laboratory
- First petascale machine for open science



The Evolution of a Jaguar



38,000x peak performance increase over 13 years

"Moore's Law" predicts 512x speedup

Science!

- Weather prediction, **climate modeling**
- Astronomy (supernova modeling, dark matter)
- Biofuel production / enzyme behavior
- Protein folding
- Efficient combustion engines
- Fusion reactor design
- Materials science (superconductors, semiconductor physics, supercapacitors)

Our Group's Challenges

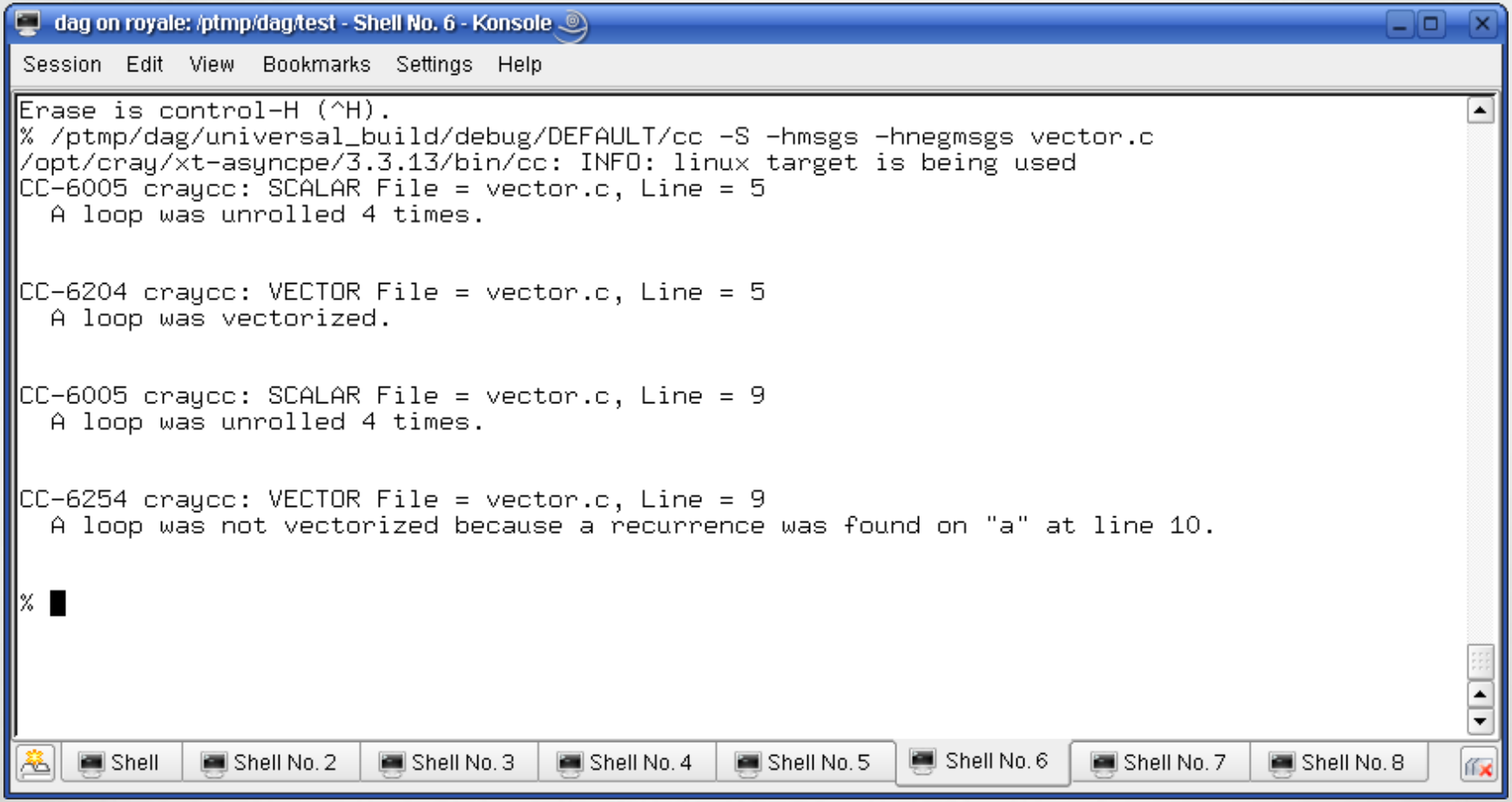
- Keeping users productive
  - Language support
  - Programming tools
  - Enormous codes
  - Feedback
- Using flops efficiently
  - Memory bandwidth
  - Vectorization & parallelization
  - Instruction selection
- Securing bid wins (rapid response)
- Keeping compiler developers sane
  - Compiler debug hooks
  - Ubiquitous IR dumps

The Compiler's Role

- LLVM is a key technology
  - Small compiler group
  - Went through an extensive internal review to justify x86 project
  - There were those who said we couldn't do it
  - LLVM made it possible! (6 months to working prototype)
- LLVM lets us
  - Keep our frontends
  - Keep our optimizer
  - Fully support Cray machines (e.g. network interfaces)
  - Rapidly respond to changing customer needs
- Optimizer (PDGCS)
  - Sits directly in front of LLVM
  - Scalar transformations, restructuring, vectorization, parallelization

Demo

Demo



Look at LLVM go!

Gyrokinetic Toroidal Code (GTC) on Jaguar

http://www.nccs.gov/2009/08/17/fusion-gets-faster

- Fusion code for ITER development
- 2x speedup over previous best (non-Cray compiler)
  - I/O & filesystem enhacements (focus of the article)
  - Compiler improvements
- Compiler contribution
  - Vectorize more than others, particularly low trip count loops
  - General memory bandwidth improvements
    - Prefetching
    - Reuse analysis (optimizer & LLVM)
  - Instruction selection improvements (Opteron 10h / Barcelona)
    - Relaxed alignment restrictions
    - Non-temporal moves

Some of What Works (The LLVM “Good”)

- Great user community
- Well-designed modular architecture
- Rock-solid (very few bugs we didn’t introduce)
- Scalar evolution!
- bugpoint
- TableGen (though see the next slide)
- Documentation (though see the next slide)

What Needs Work (The LLVM TODO List)

- A roadmap (major release goals)
- Scalability (very good, but could be better)
- Untested code paths (e.g. schedulers)
- TableGen (esoteric, missing features / multiclass support)
- Documentation (keep it up to date!)
- API fluctuation (deprecation policy)
- More microarchitecure specialization (x86 is Intel-centric right now)
  - Revision-specific instructions & features
  - Memory system information
- Debug hooks
  - Selection / schedule dags can be difficult to debug
  - Something to filter enormous amounts of debug output
  - Visualization tools

What We Plan to Contribute Near-Term

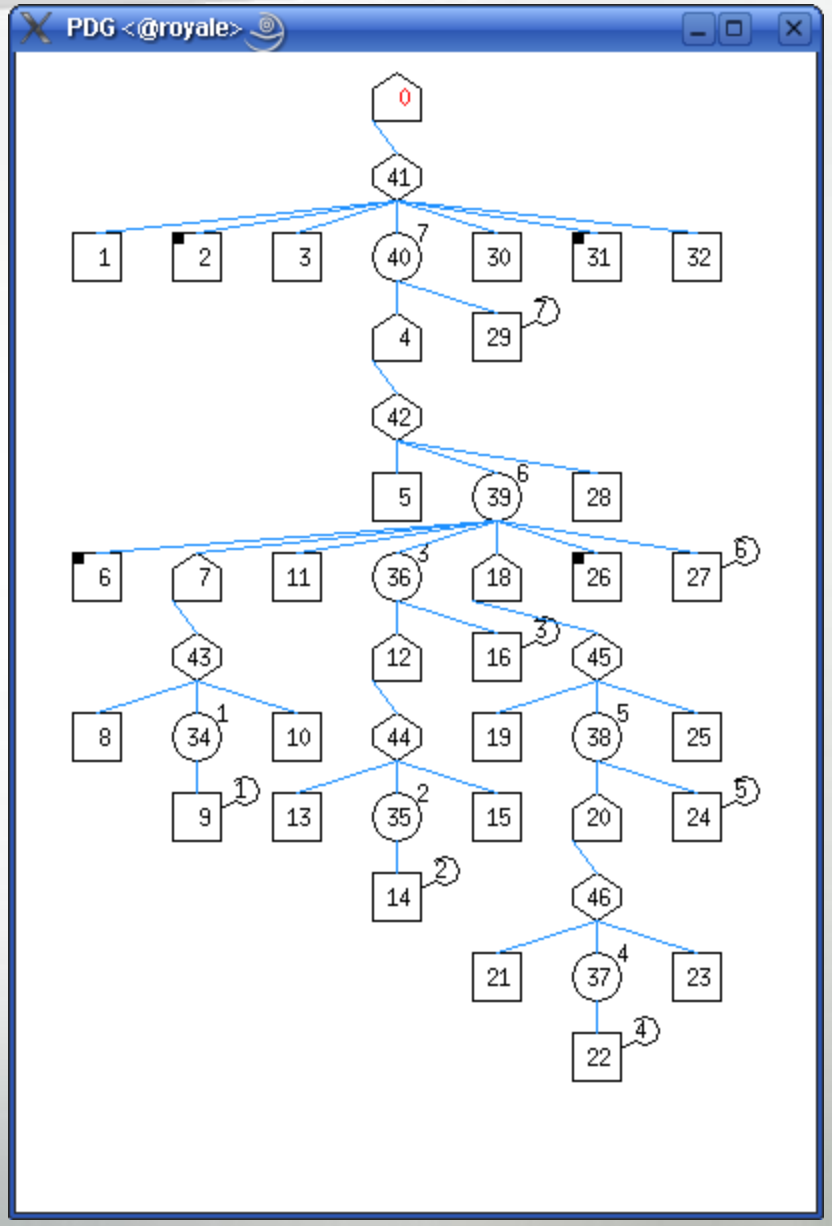
- AVX (LRBni?) implementation (including a rewrite of the SSE specification)
- Tons of debug features
  - Circular buffers
  - Before / after dumps
  - Binary search hooks (disable transformations per-function, transformations max)
  - Asm annotation
  - Enhanced bugpoint to work with compilers other than gcc (Fortran)
- Opteron enhancements (new instructions & features)
- Simple memory system models (simple!)
- Lots of bug fixes (need a solution for Fortran tests)

Looking Forward

- Near-term architectural horizon is rich!
  - GPUs
  - Larrabee (http://www.ddj.com/architect/216402188)
  - Accelerators (Cell, FPGA)
  - Manycore
- To take advantage of this LLVM should
  - Express predication (masks) in the IR
  - Include more powerful vector operations in the IR
    - Gather/scatter
    - Mixed vector/scalar

Parallelization in LLVM

- If you're writing a parallelizer
  - Provide robust messages, especially negative messages
  - Do analyses and transformations on a high-level IR (PDG or similar)
  - Drop dependence information when necessary
  - Provide a visualization of the high-level IR



Some Fun

PDGCS/LLVM

We compile so you don't have to

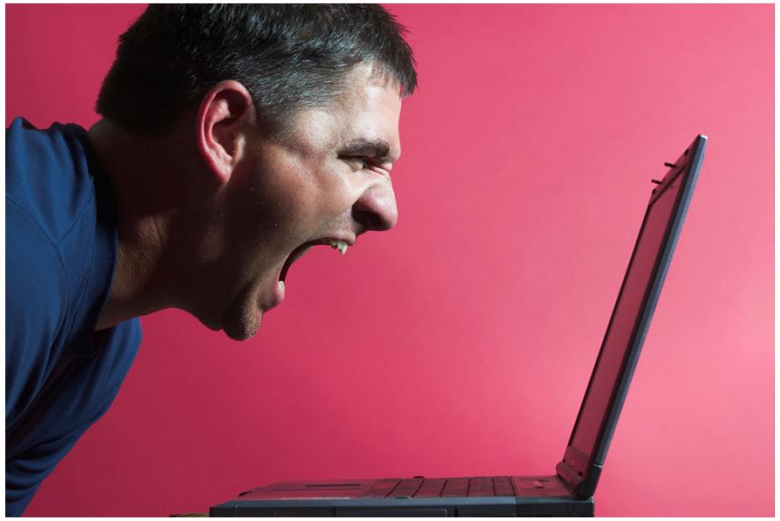
CRAY

CRAY

PDGCS/LLVM

Compiling tomorrow today

Tired of slow software?



PDGCS/LLVM

It's crack for your hack

Did I Mention We're Hiring?

http://www.cray.com/About/Careers.aspx

Q&A