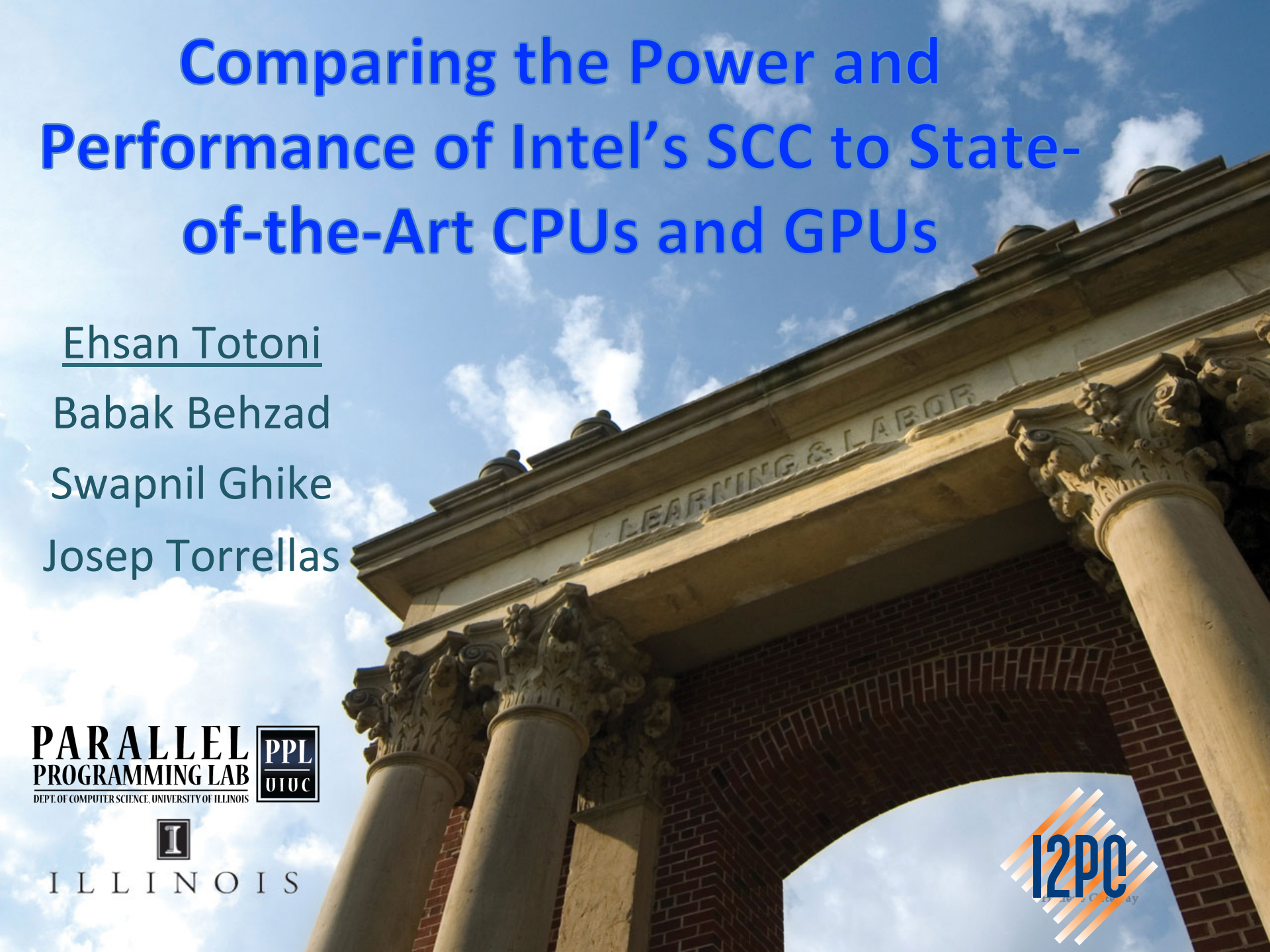
Comparing the Power and Performance of Intel's SCC to State-of-the-Art CPUs and GPUs

Ehsan Totoni

Babak Behzad

Swapnil Ghike

Josep Torrellas



PARALLEL PROGRAMMING LAB

DEPT. OF COMPUTER SCIENCE, UNIVERSITY OF ILLINOIS

PPL UIUC

ILLINOIS

I2PC

Introduction

- Increasing number of transistors on chip
  - Power and energy limited
  - Single-thread performance limited
  - => parallelism
- Many options: heavy multicore, manycore, "SIMD-like" GPUs, low-power designs
  - Intel's SCC presents manycore

Best Architecture?

- Various trade-offs
  - Performance, power and energy
  - Programmability
  - Different for each application
- Each architecture has advantages
  - No single best solution
  - (But manycore is a good option)

What do we do?

- Benchmark different parallel applications
  - On all architectures
- Analyze results, learn about architecture
- Where does each architecture stand?
  - In various tradeoffs
- What could be probably better for future?
  - Reasonable in all metrics: power, performance…
  - For most applications

Applications

- Jacobi: nearest neighbor communication
  - Standard floating-point benchmark
- NAMD: Molecular dynamics
  - Dynamic scientific application
- Nqueens: state space search
  - Integer intensive

Applications

- CG: Conjugate Gradient, from NAS Parallel Benchmarks (NPB)
  - A lot of fine-grain communication
- Integer Sort: represents commercial workloads
  - Integer intensive, heavy communication

Application Source Codes

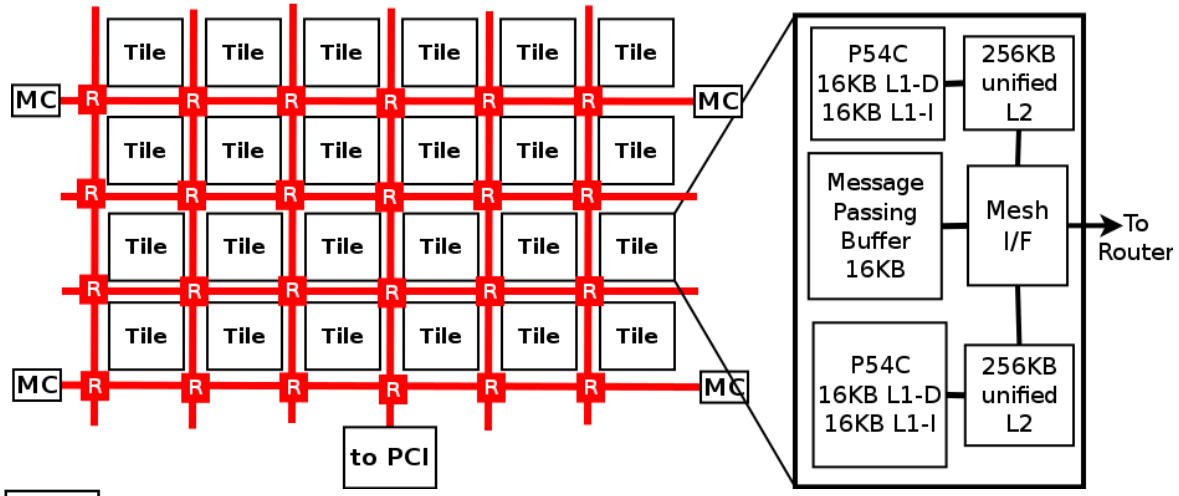
- All codes reasonably optimized, not extensively!
  - Programmer productivity
- MPI and Charm++
- CUDA and OpenCL codes for GPU
  - Reasonably optimized, same complexity as CPU ones
  - Fair apple-to-apple comparison

Platforms

- Multi-core: Intel Core i7
  - 4 cores, 8 threads, 2.8 GHz
- Low-power: Intel Atom D
  - 2 cores, 4 threads, 1.8 GHz
- "SIMD-like" GPU: Nvidia ION2
  - 16 CUDA cores, low power, 475 MHz
- Manycore: Intel's SCC
  - 48 cores, 800 MHz

Intel SCC

- "Single-Chip Cloud Computer" (SCC)
  - Just a name, not only for cloud
  - Experimental chip, for research on manycore
    - Slower than production chips, but we can analyze
  - 48 Pentium cores, Mesh interconnect
    - Not cache coherent, message-passing programs



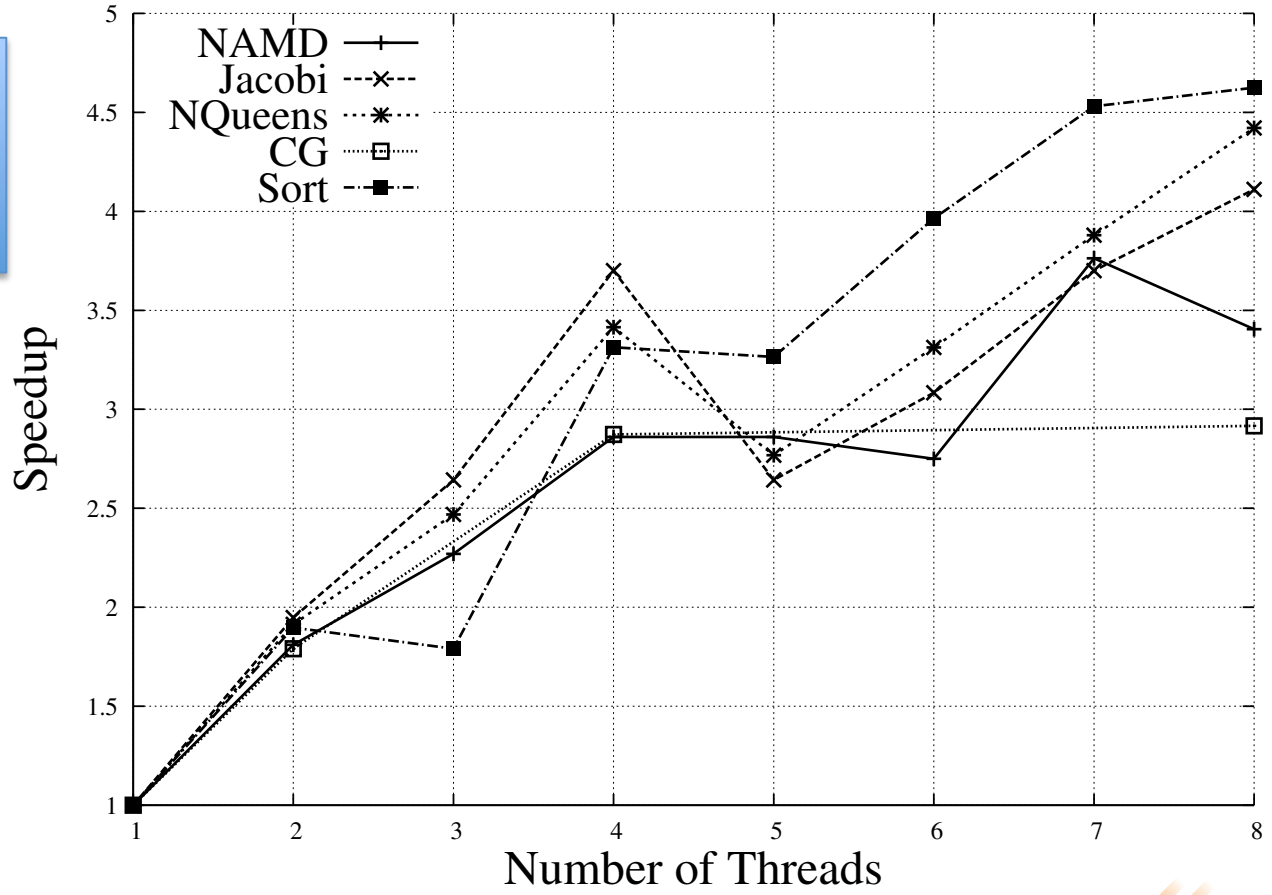
Porting to SCC

- We ported Charm infrastructure to SCC
  - To run Charm++ and MPI parallel codes
  - Charm++ is message-passing parallel objects
- No conceptual difficulty
  - SCC is similar to a small cluster
- Just headaches of experimental chip
  - Old compiler, OS and libraries…

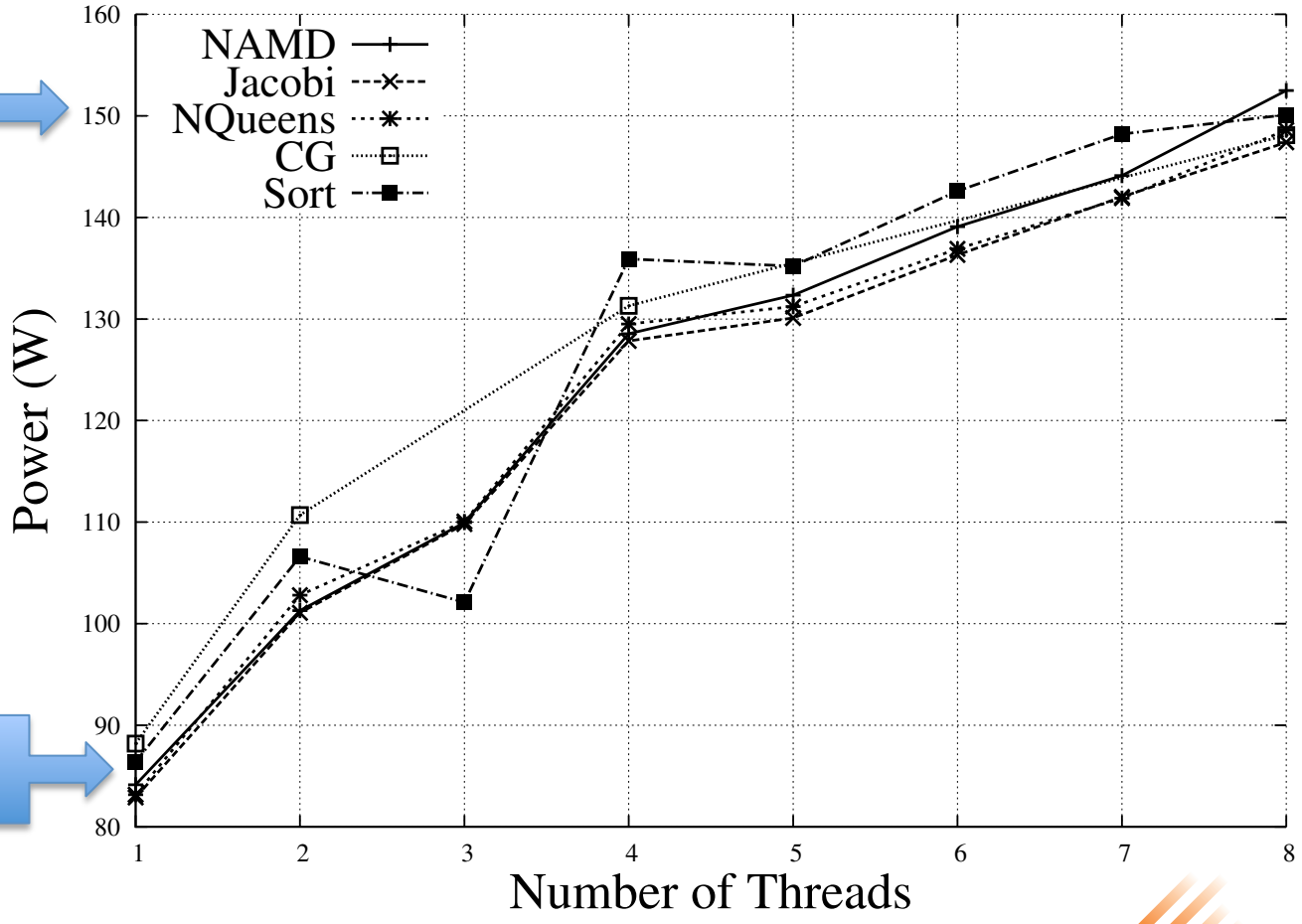
Results: Intel Core i7

Apps can use parallelism in architecture

Good scaling for all apps



Results: Intel Core i7

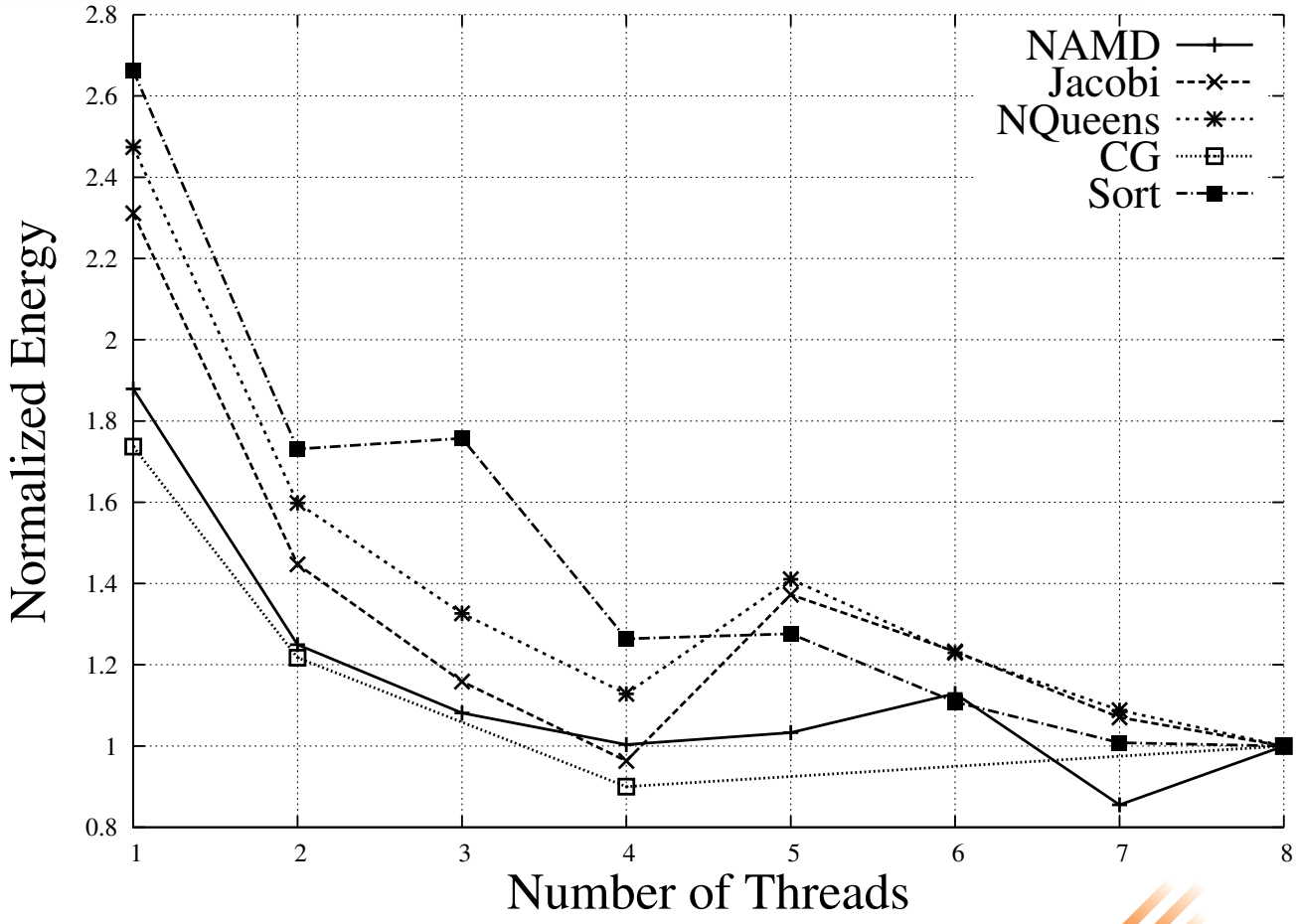


High Power increase

Results: Intel Core i7

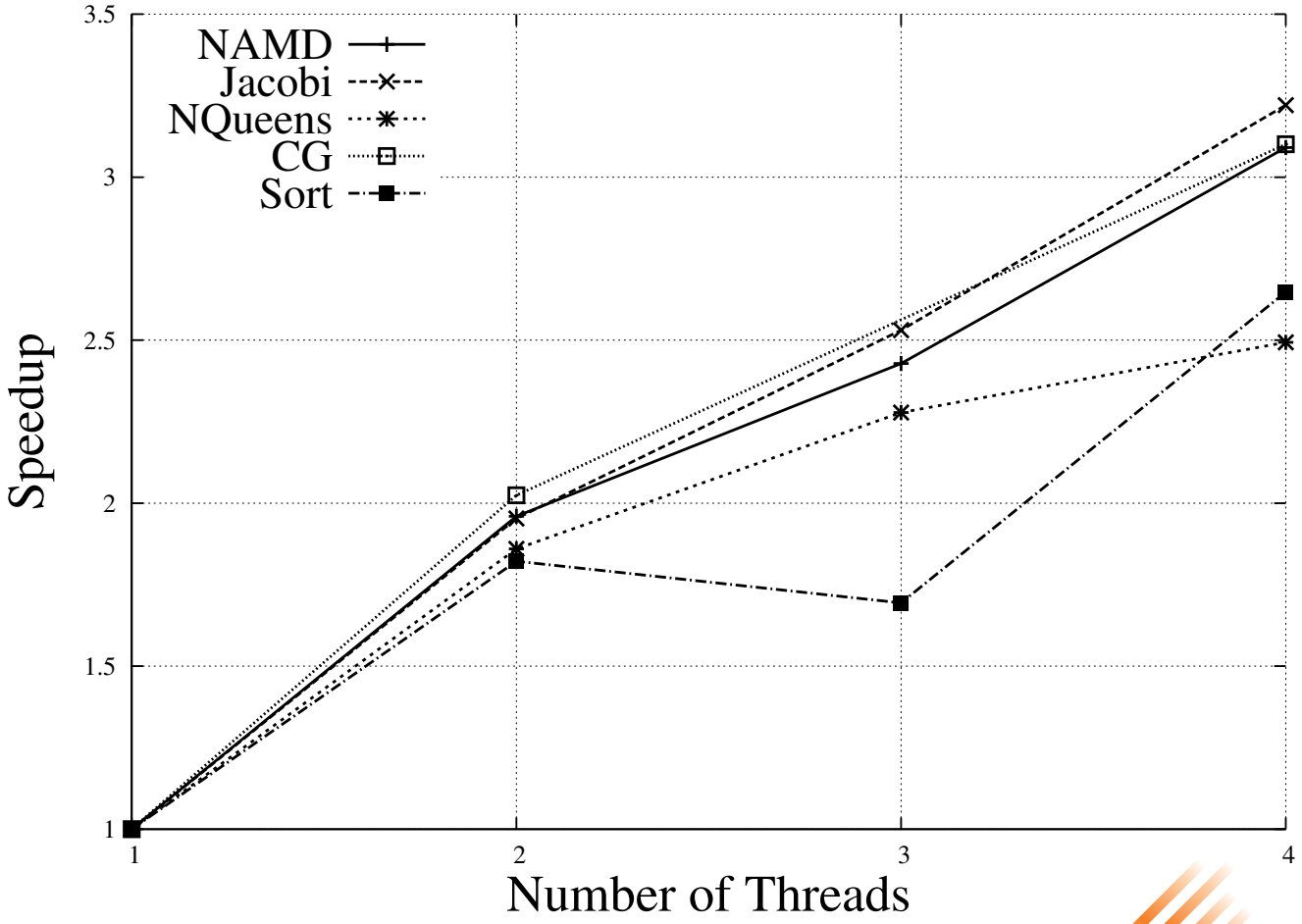
Speedup offsets
Power increase

Energy scales with parallelism



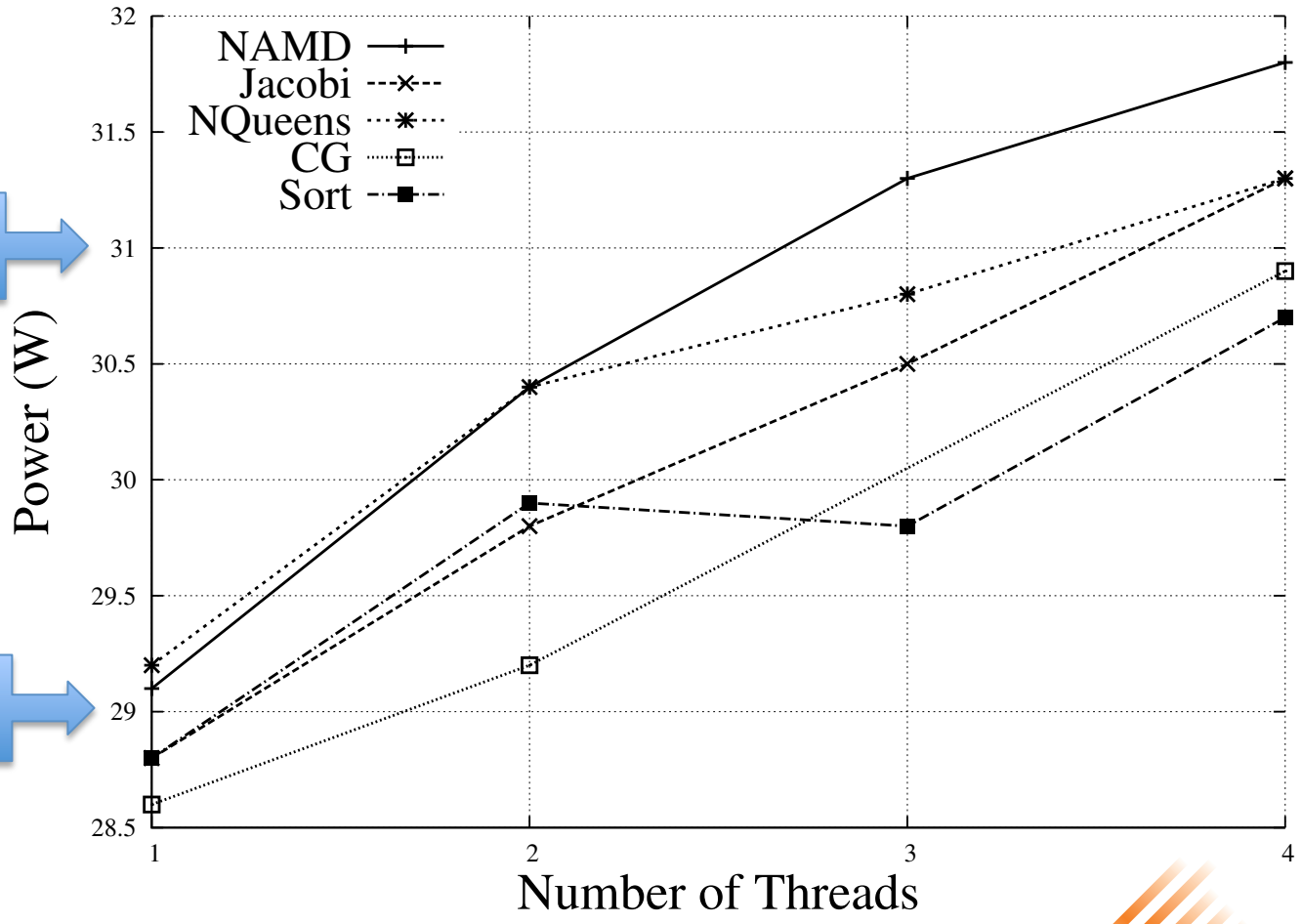
Results: Intel Atom Speed

Good scaling for apps



Results: Intel Atom Power

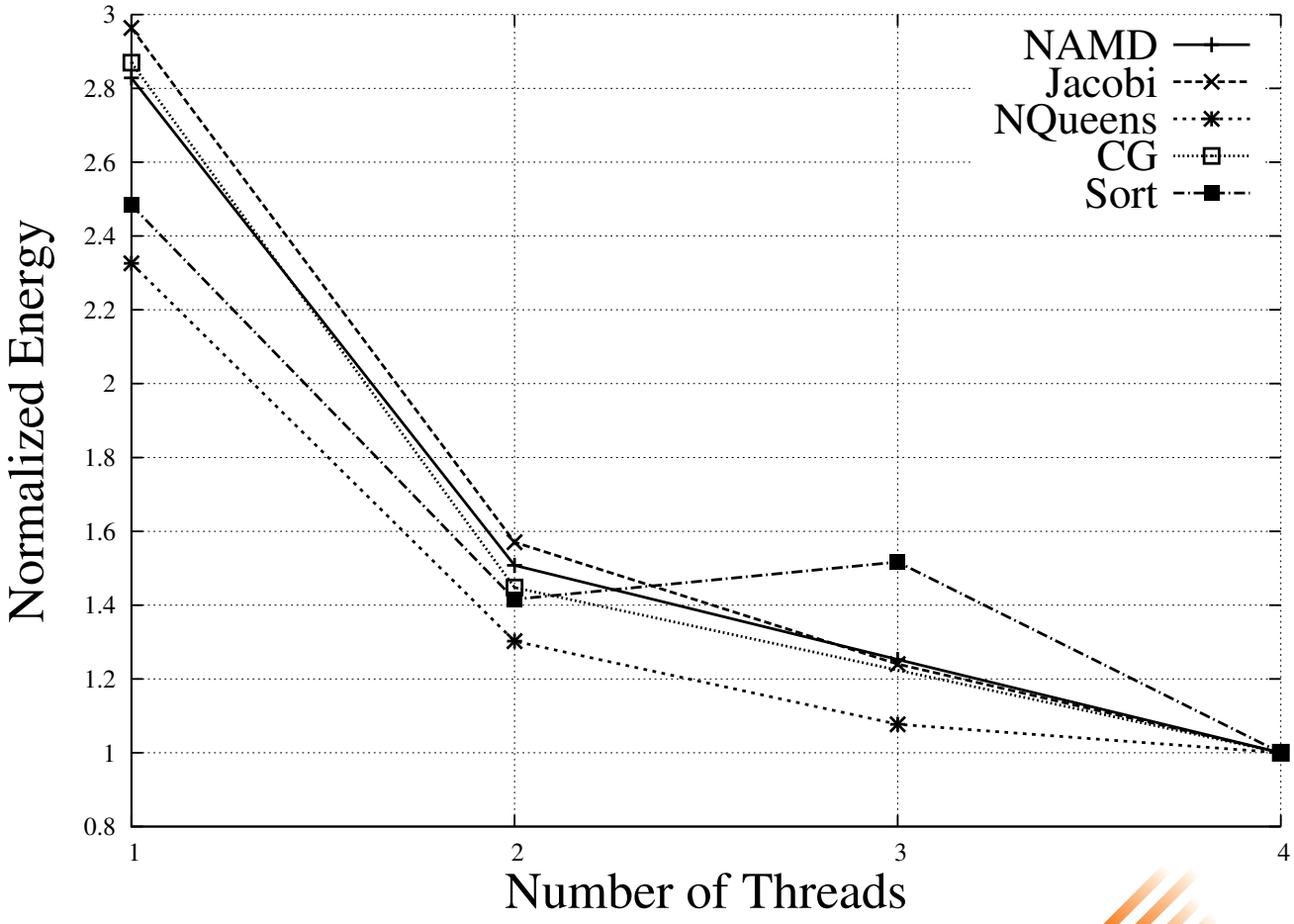
Low Power increase



Results: Intel Atom Energy

Speedup and near constant power

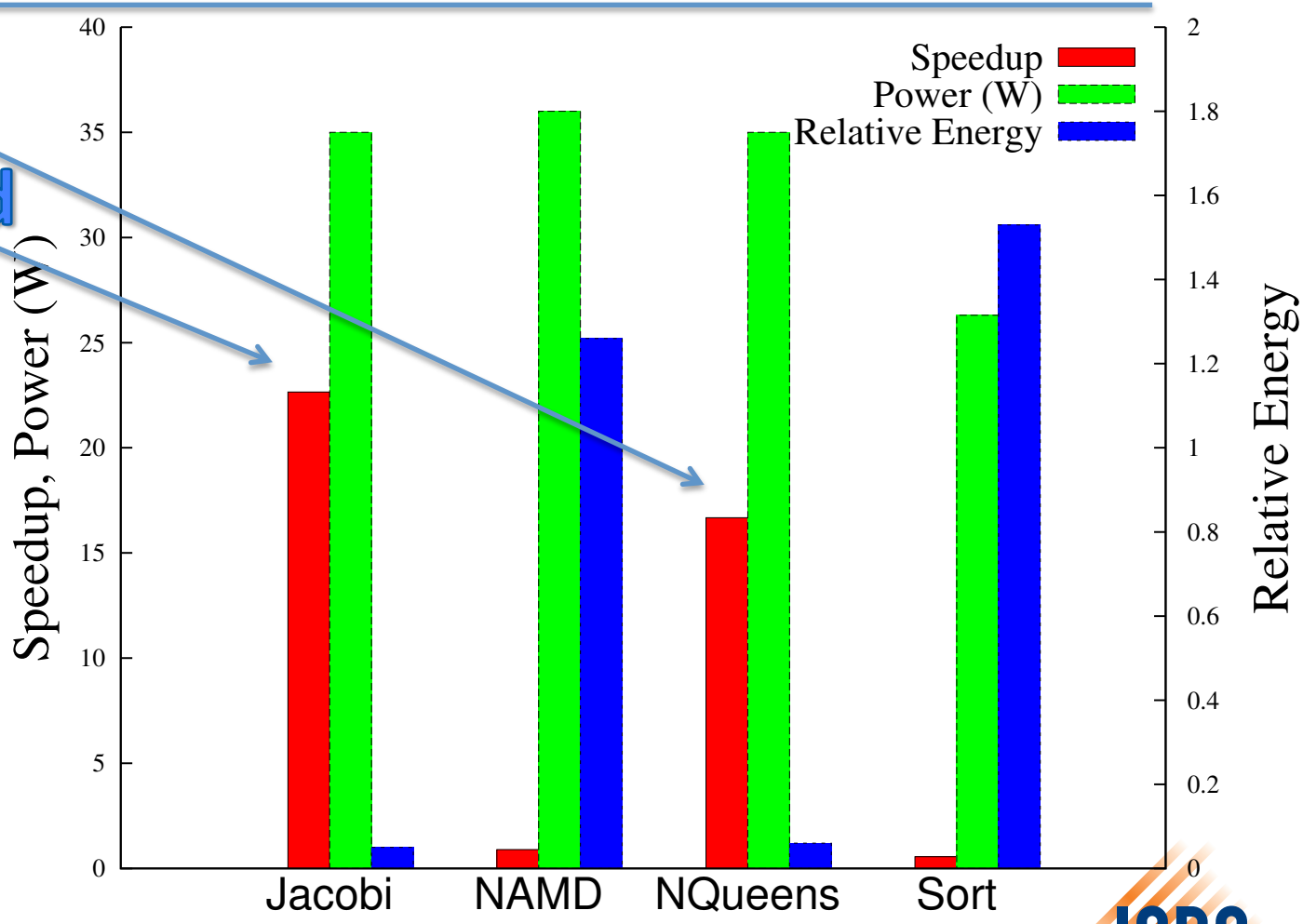
Energy scales with parallelism



Results: GPGPU

Good speedup and Energy for some apps

Low Power

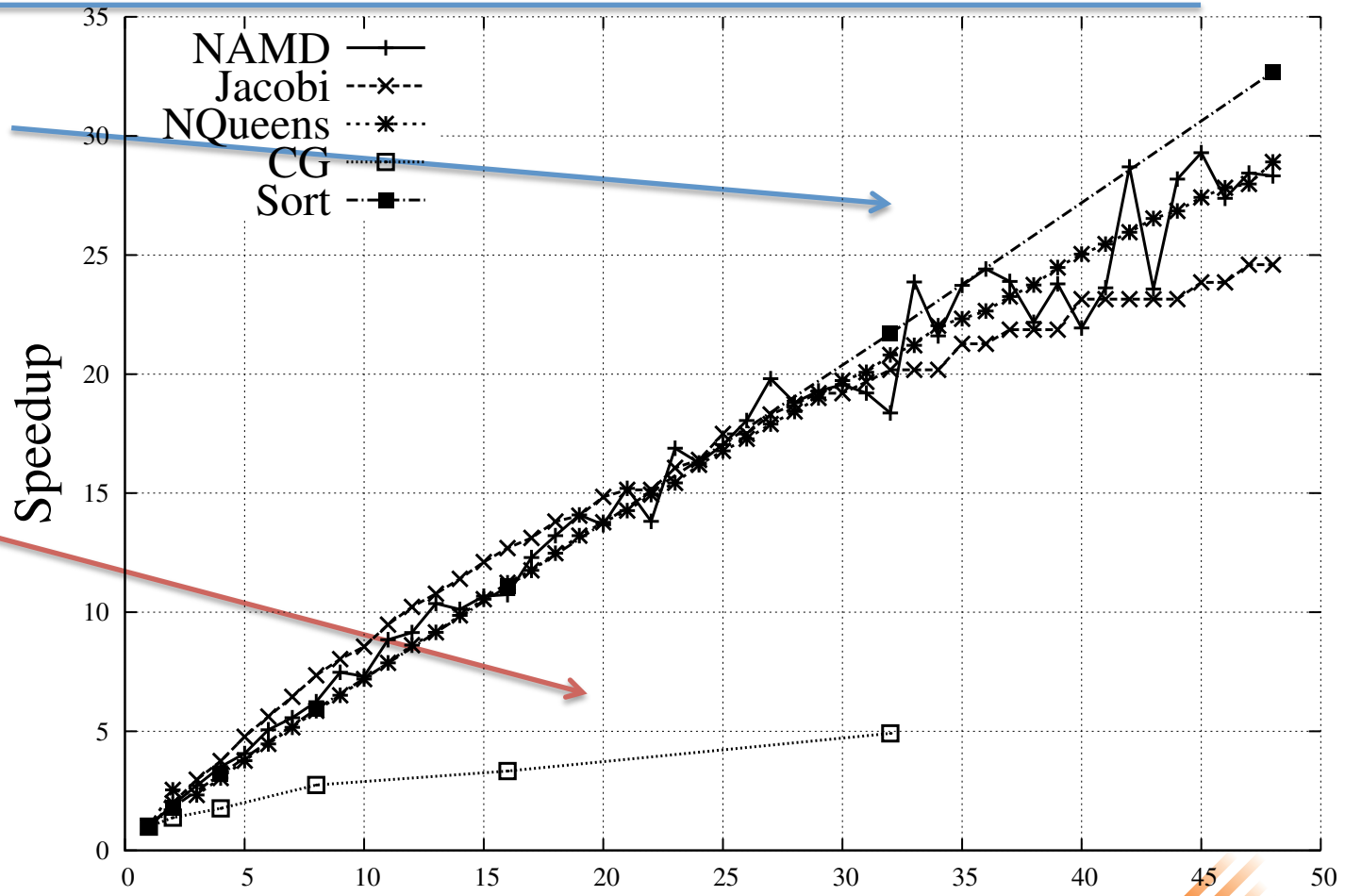


Results: Intel SCC Speed

No change in source code!

Applications scale very well

Except CG (fine-grain collectives)



Results: Intel SCC Power

Moderate Power increase

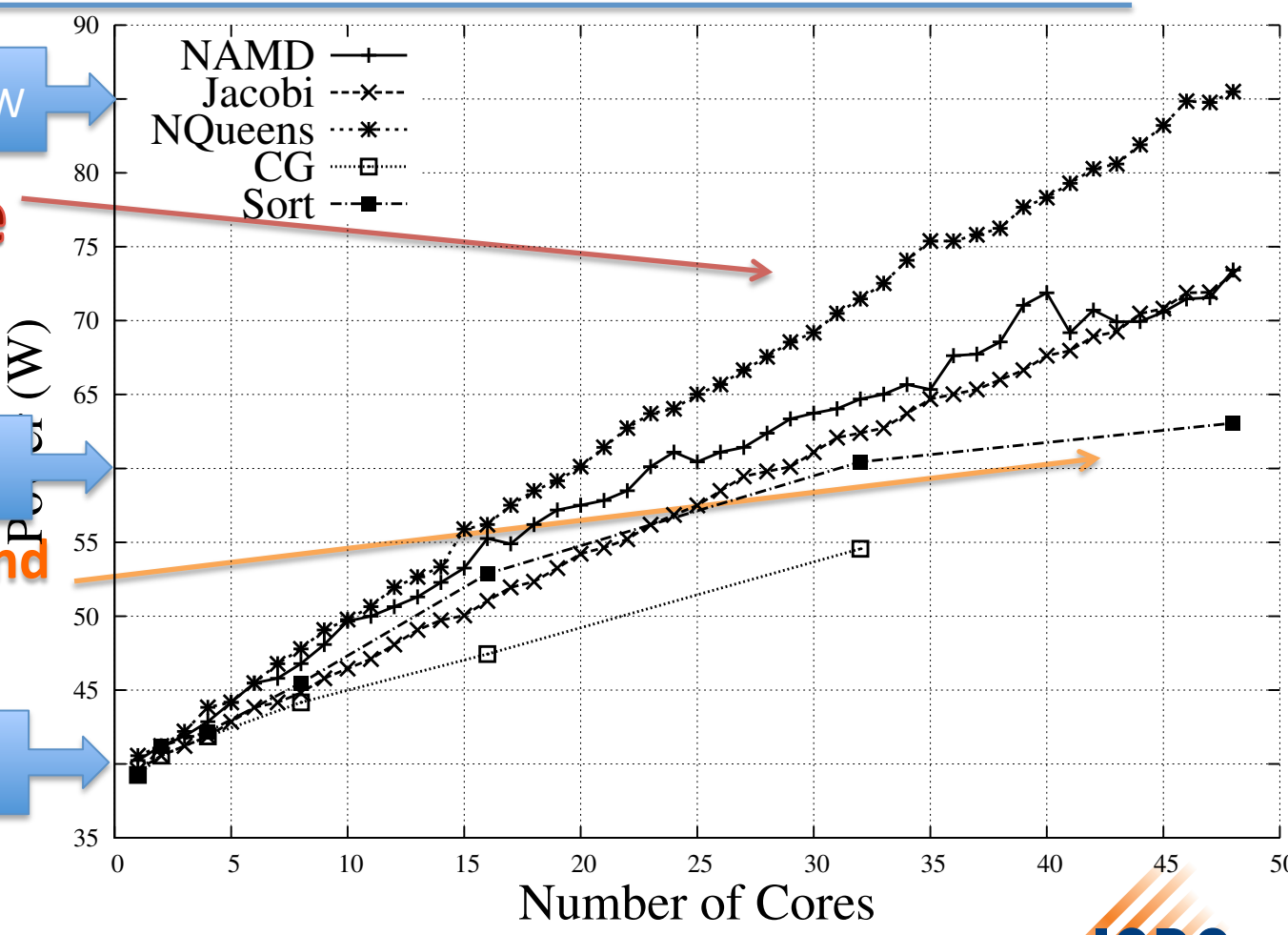
85W

Different power

60W

(Communication bound so processors stall)

40W

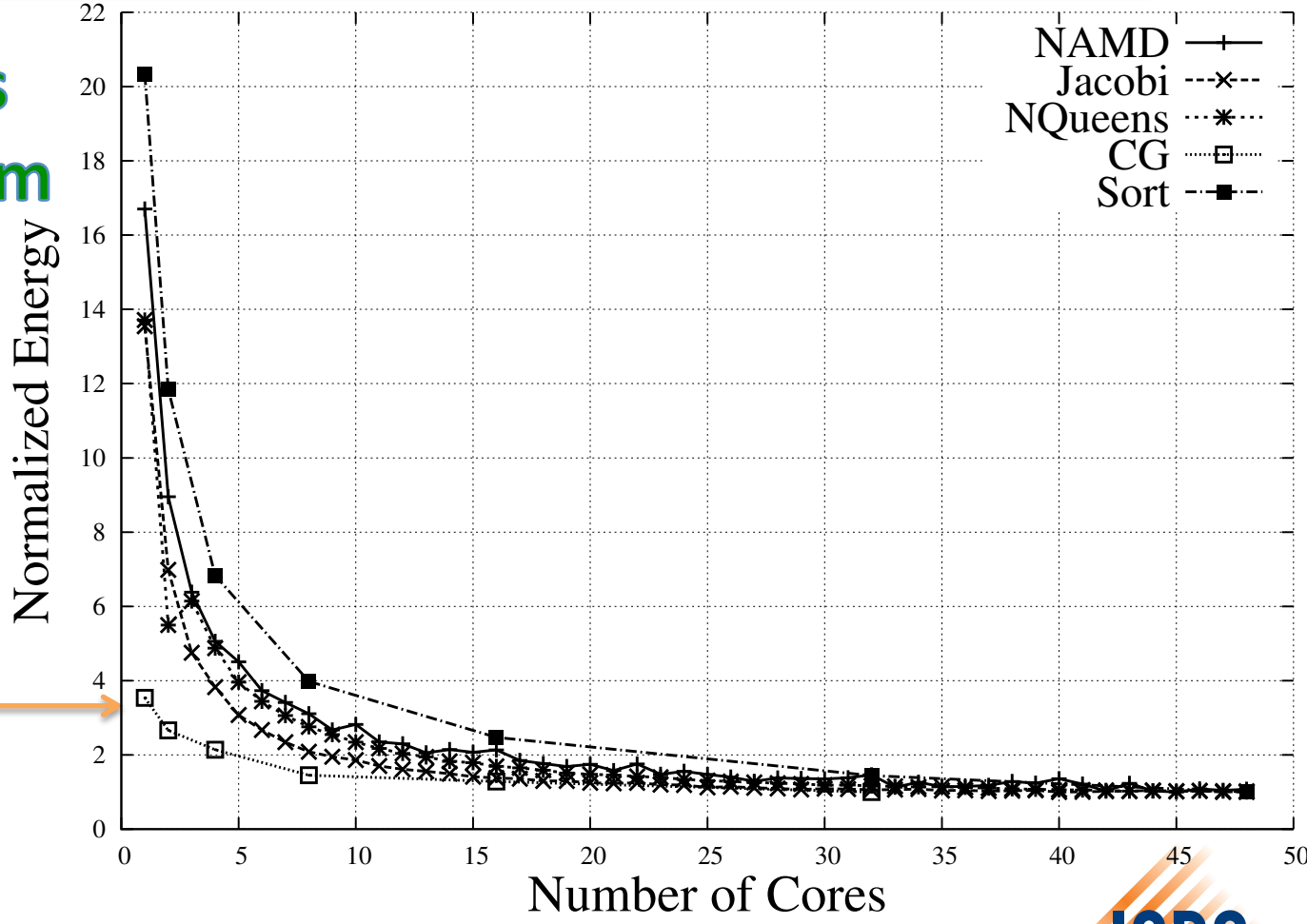


Results: Intel SCC Energy

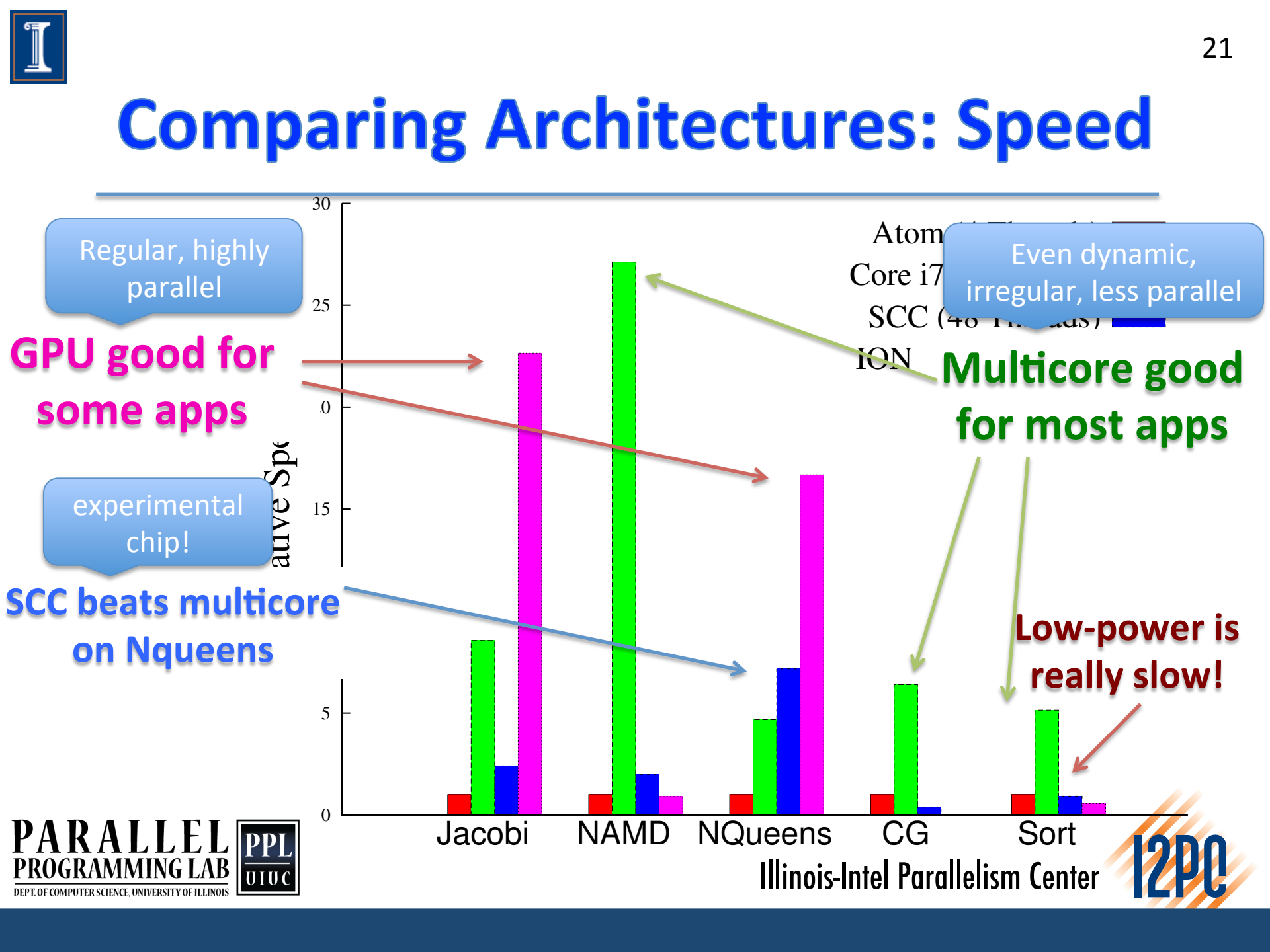
Energy scales with parallelism

Apps scalability needed to offset power increase!

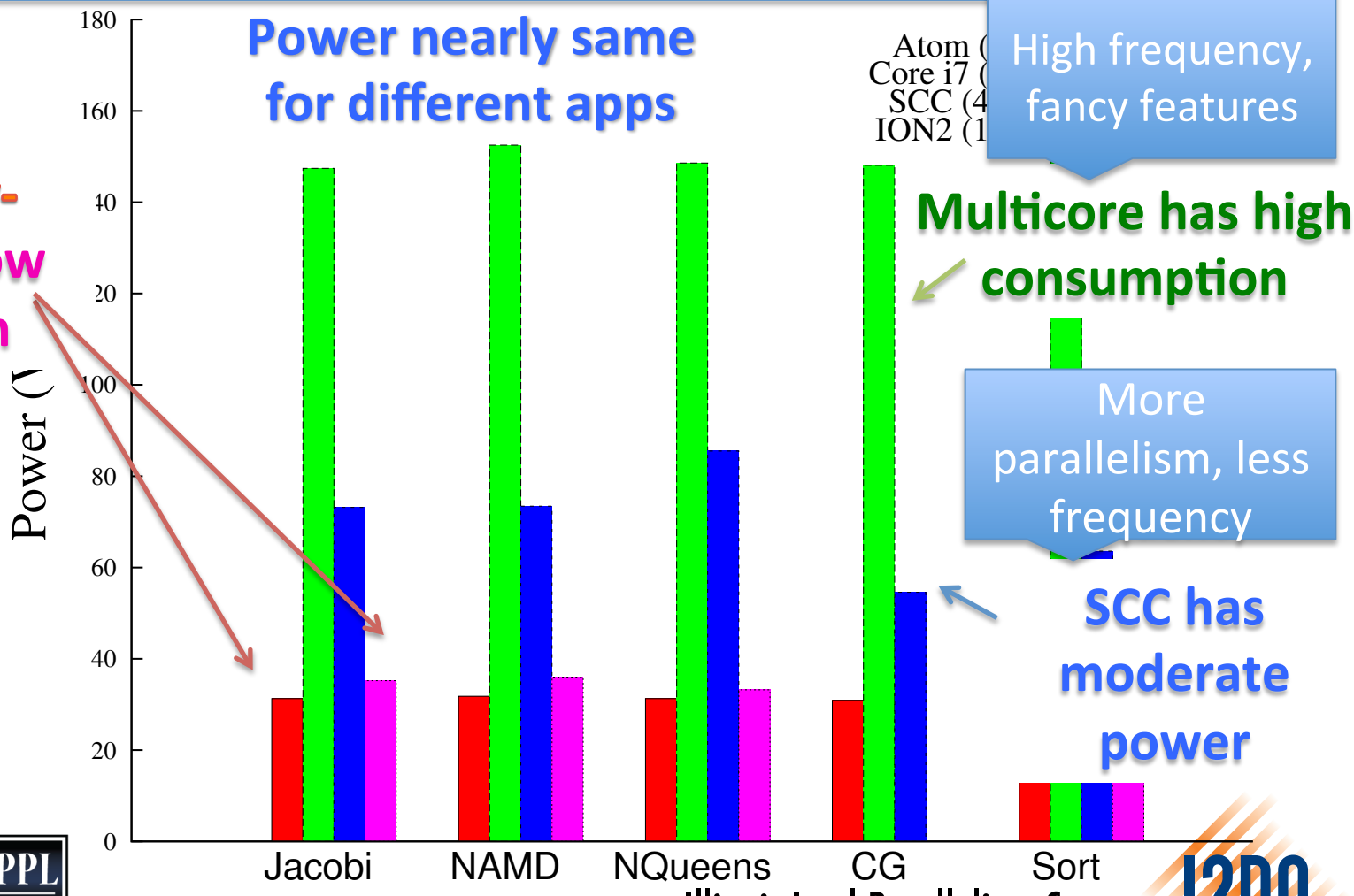
not for CG



Comparing Architectures: Speed



Comparing Architectures: Power



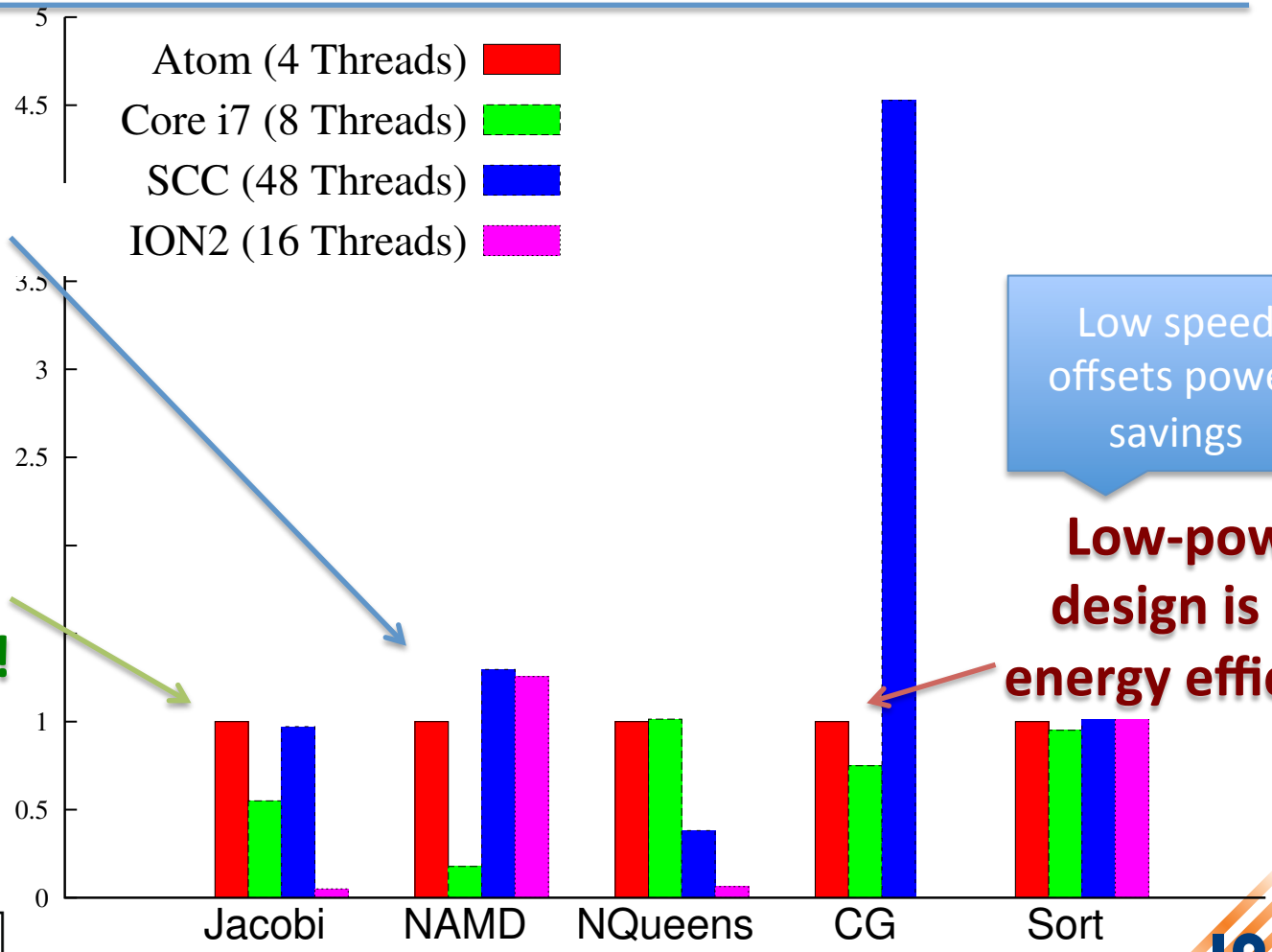
Comparing Architectures: Energy

Despite low speed

SCC is OK

High speed offsets power

Multicore is energy efficient!



Low speed offsets power savings

Low-power design is not energy efficient!

Conclusion

- Multicore still effective for many apps
  - Dynamic, irregular, heavy communication
- GPU is exceptional for some apps
  - Not for all apps
  - Programmability, portability issues
- Low-power design just low power
  - Slow! So not energy efficient

Conclusion

- SCC: a good option
  - Low power and can run legacy code fast
- Balanced design point
  - Lower power than multicore, faster than low-power design
  - No programmability and portability issues of GPU
- Experimental, slow but can be improved

SCC improvements

- Sequential performance, floating point
  - Apps show good scaling on more cores of SCC
  - Easy with more CMOS transistors
  - Interconnect also has to improve with it (still easy)
- Global collectives network
  - Like BlueGene supercomputers, low cost
  - To support fine-grain global communications
    - As in CG


Questions?
ILLINOIS
PARALLEL
PROGRAMMING LAB
DEPT. OF COMPUTER SCIENCE, UNIVERSITY OF ILLINOIS
PPL
UIUC
I2PC