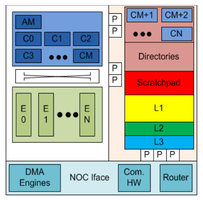
Energy-Efficient Super Computing

Stanford University

Curt Harting
Vishal Parikh
Milad Mohammadi
Tarun Pondicherry
Prof. William J. Dally

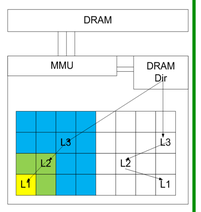
Overview

The goal of the Efficient Supercomputing Project is to significantly reduce the amount of energy consumed executing scientific code while providing programmers an API that allows for productive algorithm implementation. We do this by exposing locality to the programmer, minimize unnecessary network traffic, and reduce cache contention and meta-data overhead.



Goals

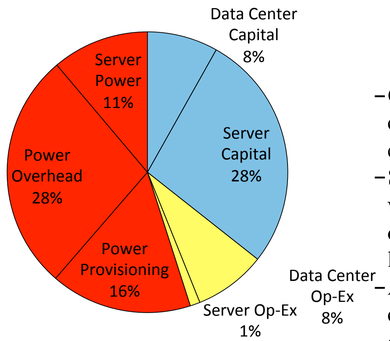
- Design a high-performance **efficient architecture,** that provides parallelism with minimal overhead over **100s-1000s of cores**
- Enable **faster, more efficient code** through software configuration of cache hierarchies and active messages
- Provide programming system**,** allowing developers to **productively implement algorithms** that optimally use hardware



Motivation

TCO of a Data Center[1]

55% due to power requirements



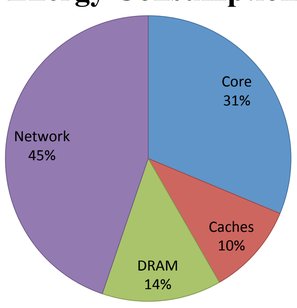
$$Power = (E_{CoreOp} * Operations_{sec}) + (E_{Comm} * Bits_{sec} * d) + (E_{Cache} * Accesses_{sec})$$

- Consumer demand for computational capabilities is increasing, while power envelopes are stationary or decreasing.
- Scaling device dimensions and supply voltage used to scale energy per operation enough, however, that is no longer the case
- Architectural innovation becomes critical in making computers more energy efficient and allowing performance to continue to grow

1. Barroso, Holzle. The Data Center as a Computer. Morgan and Claypool. 2009

Application Energy Breakdown

Splash 2 Radix Sort Energy Consumption

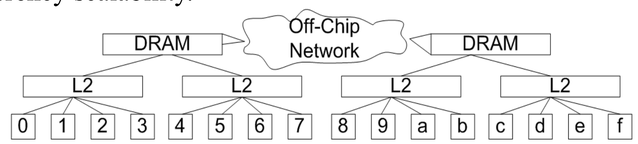


Operation	Energy (AU)
64b Integer Add	1
64b Flop	50
Read 8kB Cache	30
Route 64b on chip	160

- Data movement is 45% of the energy in a many-core radix sort, 37% in FFT, 88% in a hash table
- Cache coherent shared memory obfuscates this energy
- Improve energy-efficiency and performance in many-core processors
- Our research targets all types of energy consumption shown

Cache Hierarchy

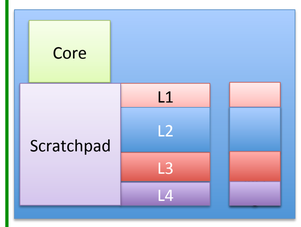
The Kapok project is focused on reducing the amount of energy consumed in the data supply on chip. Optimizing the coherent cache hierarchy is an important means by which we can do this. Novel structures and programming interfaces must be developed to improve coherency scalability.



Scalability Problems

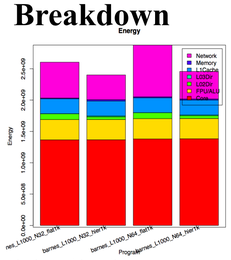
Problem	Proposed Solution/Improvement
High directory associativity	Hash based directories to improve average access latency and energy
Application Diversity	A hardware API that allows for cache configurability
Long distance miss traversals	A hierarchy of directories designed to keep miss distance and energy at a minimum.
Cache miss penalties	API designed for the programmer to be able to specify block data transfers, pinning data, and other optimizations

Configurability

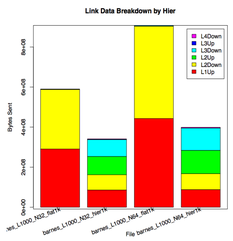


- Software configuration of cache hierarchies improves performance
- Allow the user to configure cache domains
- Provide APIs to allow for pinning data to local storage
- Convert portions of SRAM to non-coherent, locally addressed scratchpad memory

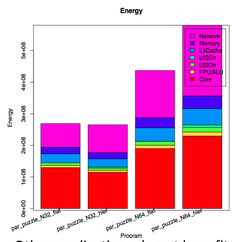
Cache Hierarchy Energy Breakdown



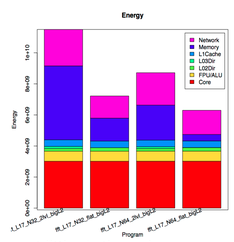
A hierarchical directory can reduce the total energy required



Energy savings are due to reduced travel distance of cache traffic

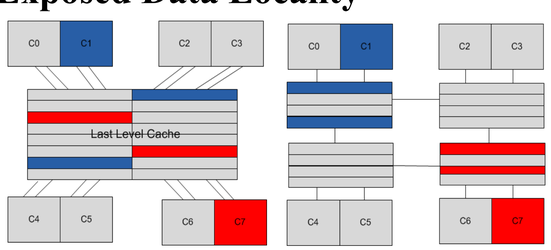


Other applications do not benefit from hierarchy, due to poor reuse or...



...not fitting in the cache

Exposed Data Locality



- Allow programmers to control data location
- Less energy spent locating and moving the data on loads and stores
- Managed either via hardware or software

Memory & Communication

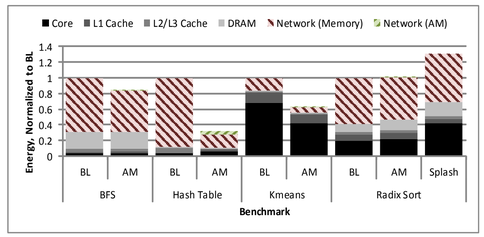
Since communication and memory energy does not scale with computational energy, data movement will become a larger problem as devices scale. Active messaging, block transfers, and fast barriers are examples of efficient communication mechanisms provided by Kapok.

Remote Messaging & Communication

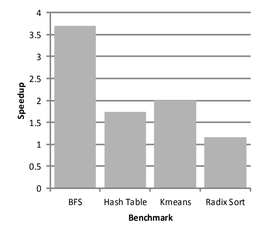
- The key to reducing the amount of energy consumed in cache coherency protocols is simple: do not miss
- Access highly contended variables/locks at their home node via active messages instead of invalidating loads and stores
- Configurable cache hierarchy allows programmers to take advantage of different forms of sharing

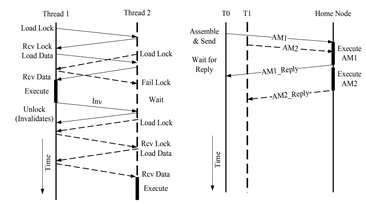
Active Messages

Energy



Speedup





Programming System

The programming system will simplify interacting with the underlying memory system without compromising configurability. Programmers should only need to focus on expressing high level intent. Syntax analysis and profiling will partially automate selecting the communication mechanism. Annotations in code will signal programmer intent.

Profiling

Memory address contended, use remote writes

Good locality, use cache

Thread A Write
Thread B Write
Thread C Write

- Profiling information used to suggest communication mechanisms
- Design compilers to automatically select communication mechanisms for programs