Energy Efficient Computing

Kapok & ELM: Reducing the energy cost of parallel computation

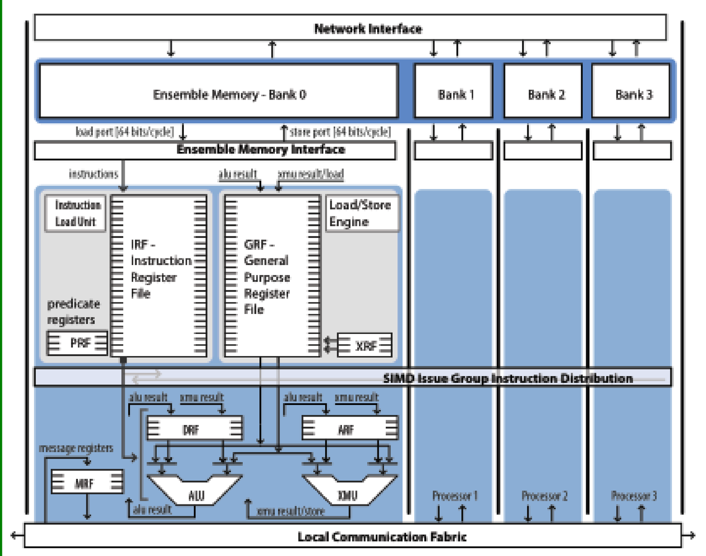
Stanford University

Professor William J. Dally
Curt Harting
Vishal Parikh
Jongsoo Park

ELM: Embedded Computing

The ELM project focuses on the creation of low-power, high-performance, programmable embedded systems. By designing systems composed of many efficient processor and memory tiles and providing complete programming and efficient run-time environments, ELM will significantly reduce or eliminate the need of fixed-function hardware in many systems.

Architectural Overview



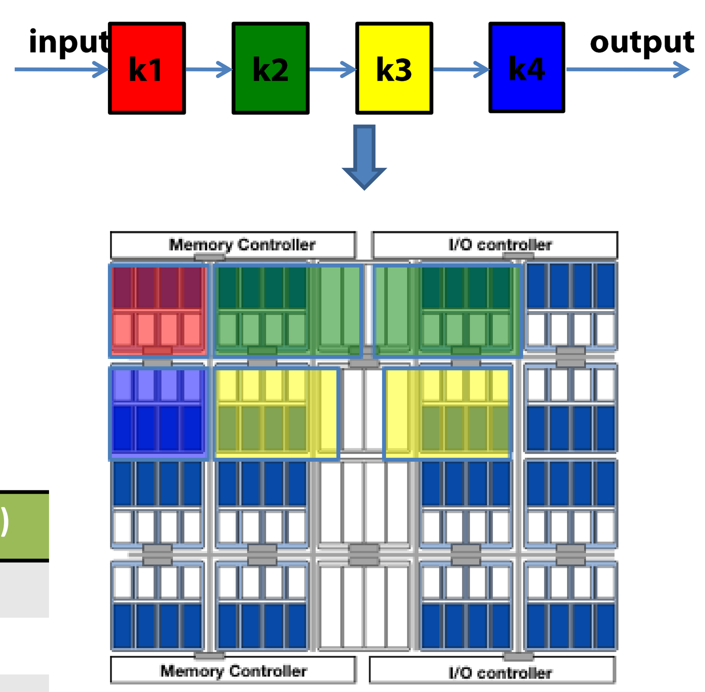
- Homogeneous tiles with 4 small processors and 8kB of software controlled memory
- Instructions from software managed instruction register file
- Data register files of varying size
- Hardware engines for data movement

Hardware Mechanisms For Embedded Software

Software Characteristic	Hardware Feature
Instruction Level Parallelism	2 Wide VLIW Ensemble Processors
Regular Data Access Patterns	Hardware Stream Engines for block transfers
Small Code Size	64 entry, software controlled instruction register file
Tight Loops	Auto-incrementing loop counters and indices into both memory and GRF
Data Level Parallelism	All 4 EPs in an Ensemble can execute in SIMD mode
Thread Level Parallelism	Many core shared address space chip-level architecture

Software Design Flow

- Developers write high level code as streams, kernels and throughput constraints
- The programming system optimizes and connects kernels
- The compiler optimizes and parallelizes low-level kernels



Transfer Type	Time (norm)
Blocking Load	1
Blocking Streams	0.52
Double Buffering Stream	0.42

Execution Time of a Histogram Program using Remote Data

Communication

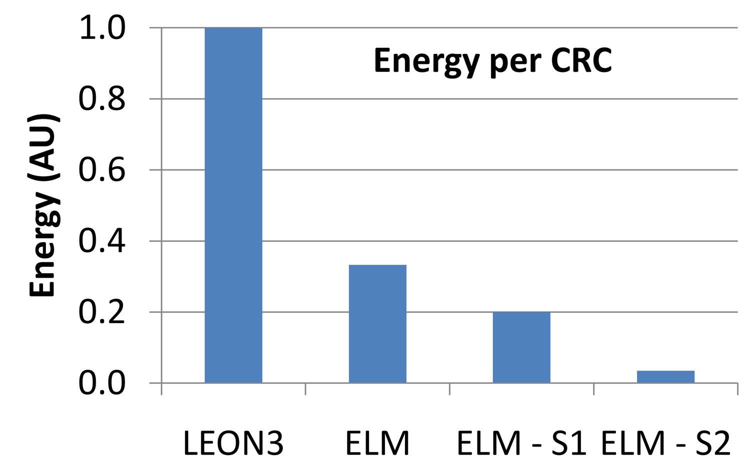
- Intra-Ensemble communication via zero overhead message registers
- Software/hardware controlled on chip memories stages data transfers
- Hardware stream descriptors and block memory transfer mechanisms provide efficient data movement

Overview

The CVA Group focuses on the creation of energy efficient, high-performance, programmable computing systems. The embedded ELM architecture is comprised of many efficient processor and memory tiles and provides complete programming and run-time environments. The Kapok supercomputing architecture focuses on scientific applications with large floating point data sets and process communication.

Goals

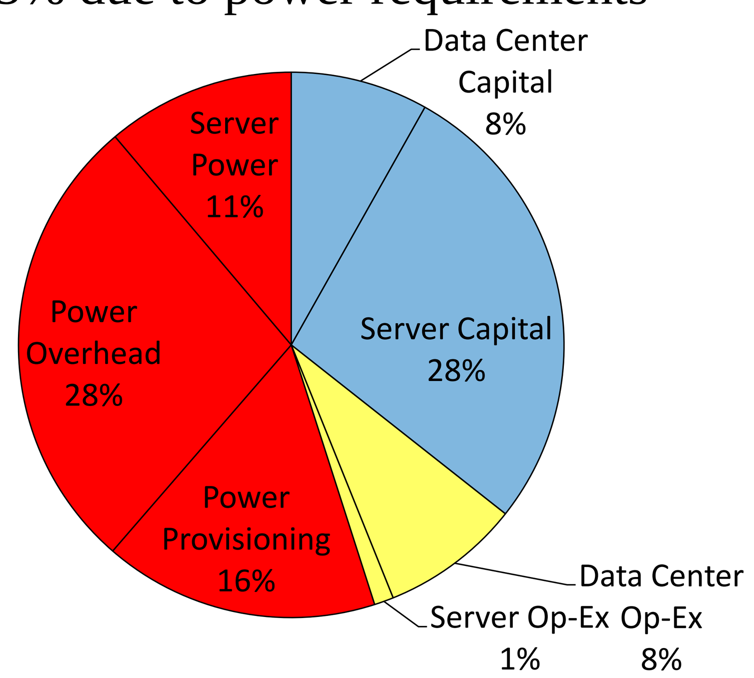
- Design high-performance **efficient architectures,** exposing fine-grained parallelism to the user
- Provide **novel mechanisms** for programmers to write faster, more efficient code
- Build a **software design environment ,** allowing developers to productively implement fine-grained parallel algorithms
- Reduce energy consumption further via **hardware implementation** of the novel programming mechanisms.



Motivation

TCO of a Data Center[1]

55% due to power requirements



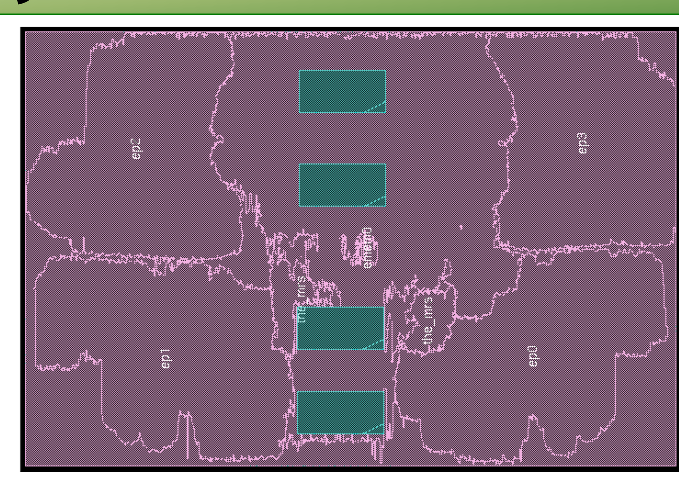
1. Barroso, Holzle. The Data Center as a Computer. Morgan and Claypool. 2009

$$Power = E_{Operation} \times \frac{Operations}{Second}$$

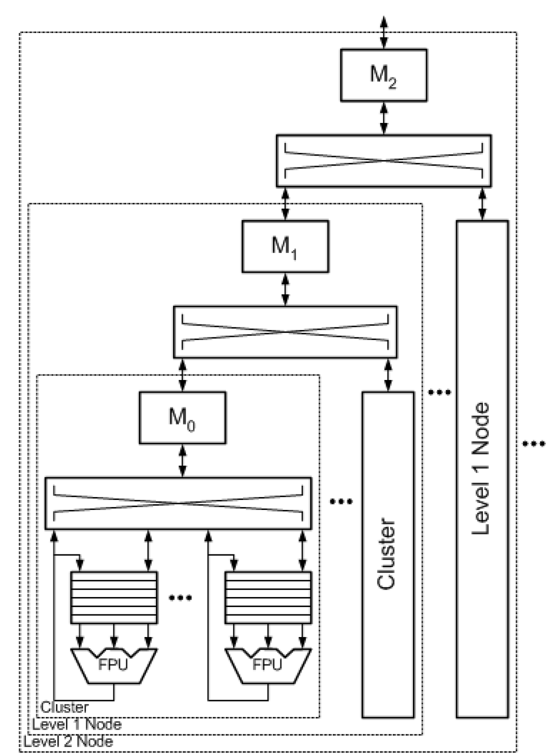
- Consumer demand for computational capabilities is increasing, while power envelopes are stationary or decreasing.
- Scaling device dimensions and supply voltage used to scale energy per operation enough, however, that is no longer the case
- Architectural innovation becomes critical in making computers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

$$Efficiency = \frac{E_{Necessary}}{E_{Necessary} + \sum_{ExtraStructures} E_s + \sum_{ExtraOperations} E_O}$$

Key Results & Future Work



- Implemented ELM's compute and memory tiles
- Demonstrated ELM to be 3-4x more efficient than conventional RISC cores
- Developed a working compiler and programming system, translating C++ to ELM assembly
- In the process of open-sourcing the RTL

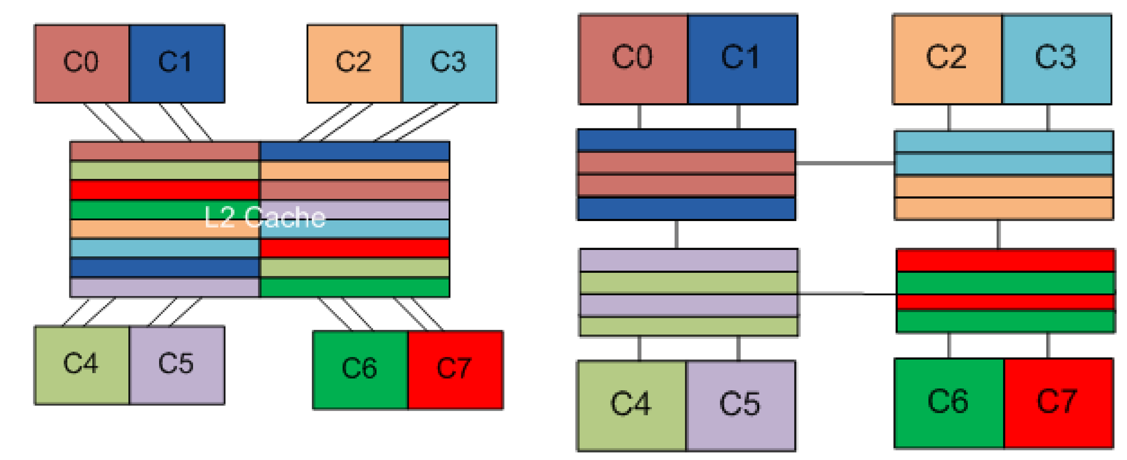


- Currently doing initial design and analysis of the Kapok memory hierarchy and messaging
- Implementing a multi-threaded simulator on top of Intel's PIN instrumentation tool
- Building Energy Models of key architectural components

Kapok: Supercomputing

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.

Exposed Data Locality



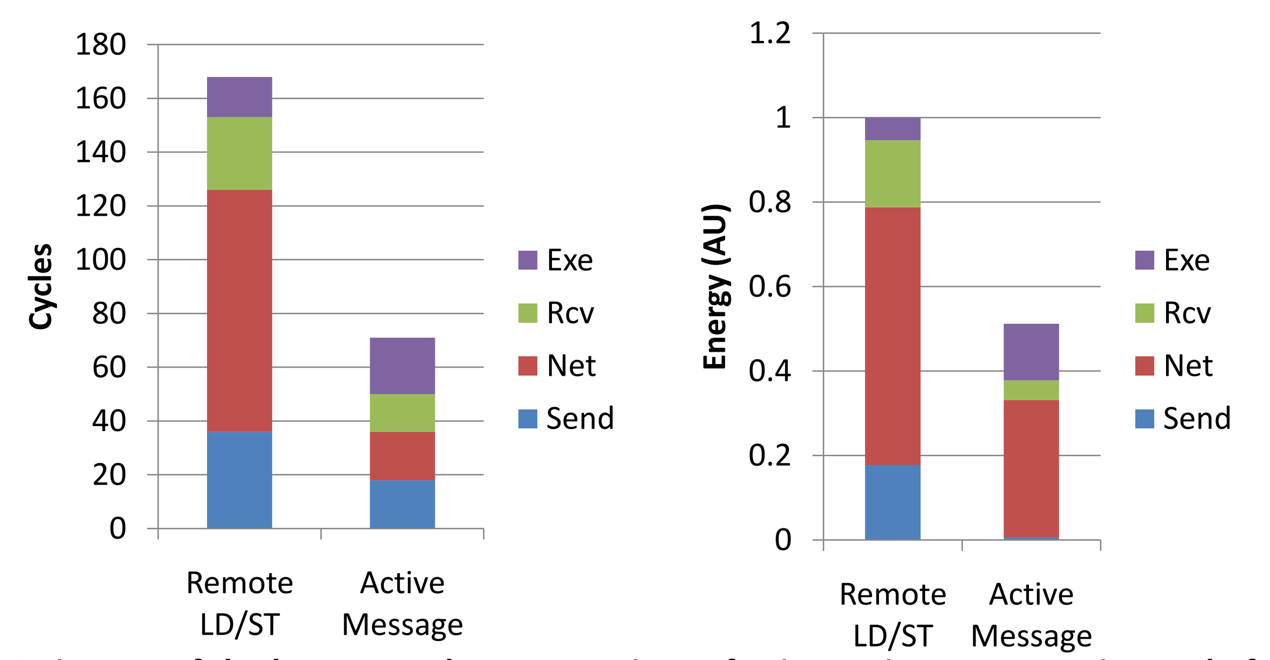
- Put data closest to the processor(s) that are currently using it
- Less energy spent locating and moving the data on loads and stores
- Managed either via hardware or software

Memory Access Patterns

Application	Pattern
Matrix Operations	Predictable blocks, suited for software
Particle Simulation	Neighborhoods, needs coalescing scatters/gathers
Graph Algorithms	Highly irregular with little locality

Remote Messaging & Communication

- Access highly contented variables/locks at their home node via active messages
- Fast barriers for decreasing synchronization overhead
- Configurable cache hierarchy allows programmers to take advantage of different forms of sharing



Estimates of the latency and energy savings of using active messages instead of loads and stores