Graph Algorithm Processor

Scott Beamer, David Patterson, Krste Asanović

EECS Electrical Engineering and Computer Sciences

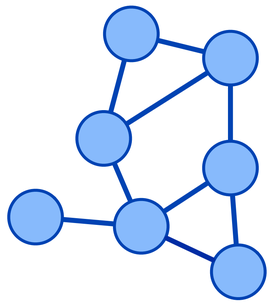
Berkeley Par Lab

PARALLEL COMPUTING LABORATORY

INTRODUCTION

Why Graph Algorithms?

- There are emerging applications that operate on large graphs (millions of nodes)
- Lots of real world data/problems record *relations*
- Some problem types:
  - Massive Social Networks
  - Scientific Data Analysis
  - Simulation
- Some of these applications can have time constraints
- Unfortunately, these can be hard for current infrastructure



Why are Graphs Algorithms Hard?

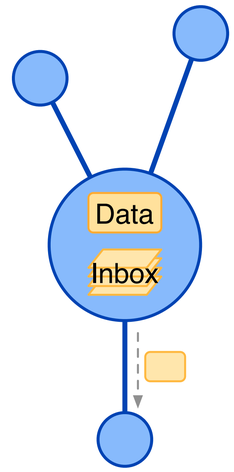
- Can often have little locality (spatial or temporal)
- Low arithmetic intensity
- Above causes memory system to become a bottleneck
  - Either bandwidth or number of outstanding requests
- Processor idles, reducing overall energy-efficiency

Why a Hardware Accelerator?

- With energy scaling slowing down, transistors are not getting much more energy-efficient, however, Moore's Law continues to give us more of them
- With a power budget, this means a decreasing percentage of the chip can be active (*Dark Silicon*)
- It follows that the active portion should be specialized for the current task
- Doing so allows us to increase its energy-efficiency to increase performance under a power budget

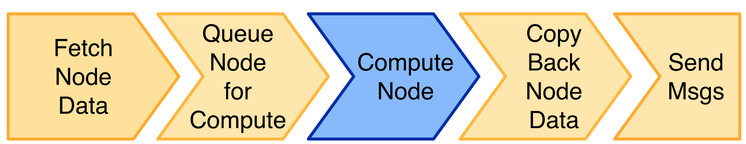
PREGEL MODEL

- *Pregel* is a Large-Scale Graph Processing Framework
  - Developed by Google, published in SIGMOD 2010
- It is Bulk Synchronous Parallel (BSP) with a "thread" per node
- Each time step, a node may:
  - Examine its private data
  - Read messages sent to it in previous time steps
  - Update its private data
  - Send messages to other nodes
- Nodes can also go to sleep until woken up by a message
- Designed to work across a large cluster

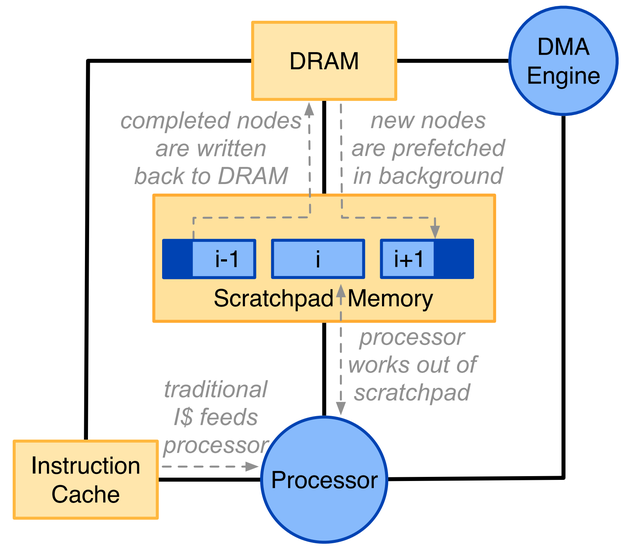


PROPOSED DESIGN

- ***Big Idea:*** Use a Pregel-like programming model to get a predictable memory accesses pattern to prefetch
  - To compute, a node only needs its data and its inbox
- Data transfers can be done asynchronously with DMA
- Can get needed parallelism without lots of threads
- Use a Scratchpad Memory (SW-managed cache) to stage data
- *Below:* Work for one node for one time step



FIRST IMPLEMENTATION



OPEN QUESTIONS

- How to layout message queues in memory
- How well could this perform on a contemporary processor?
  - Compare performance on Nehalem vs. Niagara
- Could this be done efficiently with a current processor and a Virtual Local Store?
- Processor - DMA Engine interface
- ISA extensions to assist with graphs
- Multithread the processor?
- Multiple processors?
- What other types of applications could this run?

Poster template inspired by Sam Williams et al.