# DSL Group (#1)

- Kunle Olukotun
- Jim Larus
- Krste Asanovic
- Almadena Chtchelkanova
- Mark Oskin

# Nature of applications

- Application mobility
  - Smooth transition between laptop, phone, office, coffee house
    - Security, trust
  - Rich input/output 3D display, gesture recognition
  - Adaptable to resource differentiation

# What are apps like?

- Big data crunch
- Ubiquitous machine learning
- More about enabling community
- Modeling (physics, people/economic dynamics)
- Augmented & virtual reality

# Platforms

- Mobile phone
  - Per-volume leader
- Laptop/desktop
- Cloud
- TV
- Car
  - Special regulatory requirements

## Design forces

- Energy
- Scaling
  - 10T processors in the world
  - 1TB non-volatile memory
  - 10B people
  - 1K cores per app
  - 100 different device instances
  - 10 different device classes
  - 1nm CMOS?
- Distributiveness

## Funding focus?

- Problems of commerce
- Problems of society
  - science
  - health
  - education
- They potentially converge

## How to publish in ISCA in 2020

- Realistically, ISCA will remain a simulator's game
- Can the most interesting results come from elsewhere?
  - Needs solutions to another grand challenge: Enabling $10K 1nm silicon prototypes
  - FPGA's are a fallback
  - Shared infrastructure for FPGAs and system software

## Missing Agendas

- Communication - Seems a second class system in this group
  - on-chip, on-system, across-room, across-globe
- Storage - Significant changes afoot
  - An "infinite" persistent memory?
- Kunle says what about non-Von Nuemann computing?

# DSL

- A DSL enables a developer to pass more information to the system stack and architecture
- A DSL constrains the developer expression so that it limits what problems you have to solve
- DSLs have been successful: proof is in the pudding: OpenGL (Jim says DirectX), SQL, Matlab

# Why is DSL in an architecture report?

- DSL's enable architects to find efficiencies
- Vertically integrated thinking
- Sounds nice, but designing a language is not child's play
  - Do we know what we are doing?
    - No! So embed in existing modern languages

# DSL Roadmap

- 5 years:
  - Better support in general purpose languages: multistage computation, supportive semantics
  - Environment for the washed masses to create their own DSLs
  - 2-3 prototypes of successful DSL's
    - working with specific application domains
- 10 years:
  - A commercial environment to create DSLs
  - Push-button hardware from DSL
  - 10 prototypes of successful DSL's
- 15 years:
  - 5 "as successful as SQL" languages

# Successful DSL Attributes

- Provides an abstraction the developer is comfortable with
  - e.g. sequential
  - Parallelism at a "natural" level (e.g. SQL)
- Performance that makes it worth it
- Available tools

- Measure success by adoption, a useless beautiful language is a failure

## Random Thoughts

- Translation, protection needs a do-over
- New ideas not always amendable to old SW

## DSL Group #1 - Day 2

- Mark Oskin

## Summary

- Research into domain specific languages (DSLs) has the potential to provide scalable performance and significant programmer efficiencies
- Requires a multidisciplinary collaboration between language designers, compilers, architects, and application domain experts

## Motivation

- Enable scalable application performance via an efficient translation of Moore's Law into viable architectures
- DSLs have been vastly successful in the past at enabling scalable performance and increased programmer productivity
- Majority of application domains not yet DSL'ified

# Goals

- When any developer (from expert to novice)writes for application area X they turn to a widely popular, commercially supported, DSL for X first.
- Available competing hardware solutions for X providing scalable performance, suitable for mobile - datacenter (if applicable) computing.
- DSL for area X enables scalable performance gains for the foreseeable future.

# Research Agenda: Identifying DSL apps

- Many candidate application domains
  - Machine learning
  - Speech recognition
  - Imagining
  - Graph-based computations
  - Media processing
    - currently trying to shove into GPUs

# Research Agenda: Software

- DSL for creating DSLs
  - No sense starting from scratch each time
    - Cleaner semantics if done once
    - Enables infrastructure re-use
  - Requires close collaboration between language designers and compiler writers
- Compiler infrastructure enabling DSL research
  - Target for multicore
  - Easily retargetable backend

# Research Agenda: New DSLs

- Given an application domain
  - Work closely with domain-experts, architects, and compiler writers to create DSL and supporting architecture(s)
- Iterative process:
  - Understand domain
  - Design language
  - Write application and compiler backend
  - Design architecture
  - Test
  - Repeat

## Research Agenda: UI

- Front end suitable for DSL creation and use
  - Work with HCI and SE communities
  - Focus on easy DSL design and use

## Research Agenda: Architecture

- Exploit semantics of DSL(s) to create efficient architectures
  - Must provide 100X improvement (performance X power) over multicore
  - Performance must scale with Moore's law
- Understanding a single DSL in-depth and produce best-possible hardware
  - Work across layers seeking refinement
  - Innovation across processing, communication and storage architectures
- Look for commonalities across DSLs for synthesized architectures
- Need platform for experimentation and design
  - Must be efficient-enough on multicore from a programmers perspective
    - drive adoption from software
  - Must be simple enough to experiment with on FPGAs/simple ASICs
    - ground architecture research in reality

## Measurable Outcomes


5 "As successful as SQL" DSL's
A healthy marketplace of competing solutions
1-2 significant architecture platforms

1-2 "As successful as SQL" DSL's
15 viable research DSLs covering 5-10 application areas
5-10 ongoing architecture research efforts
1-2 auto-synthesis "push button architecture generator" efforts

3-4 Research DSLs efforts covering 1-3 application areas
1-2 architecture acceleration research efforts