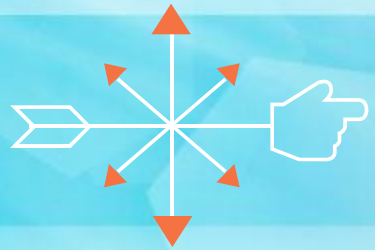
Qualcomm Research
Silicon Valley



Power ~~Parallel~~ Programming for Mobile Devices

Calin Cascaval, cascaval@qti.qualcomm.com

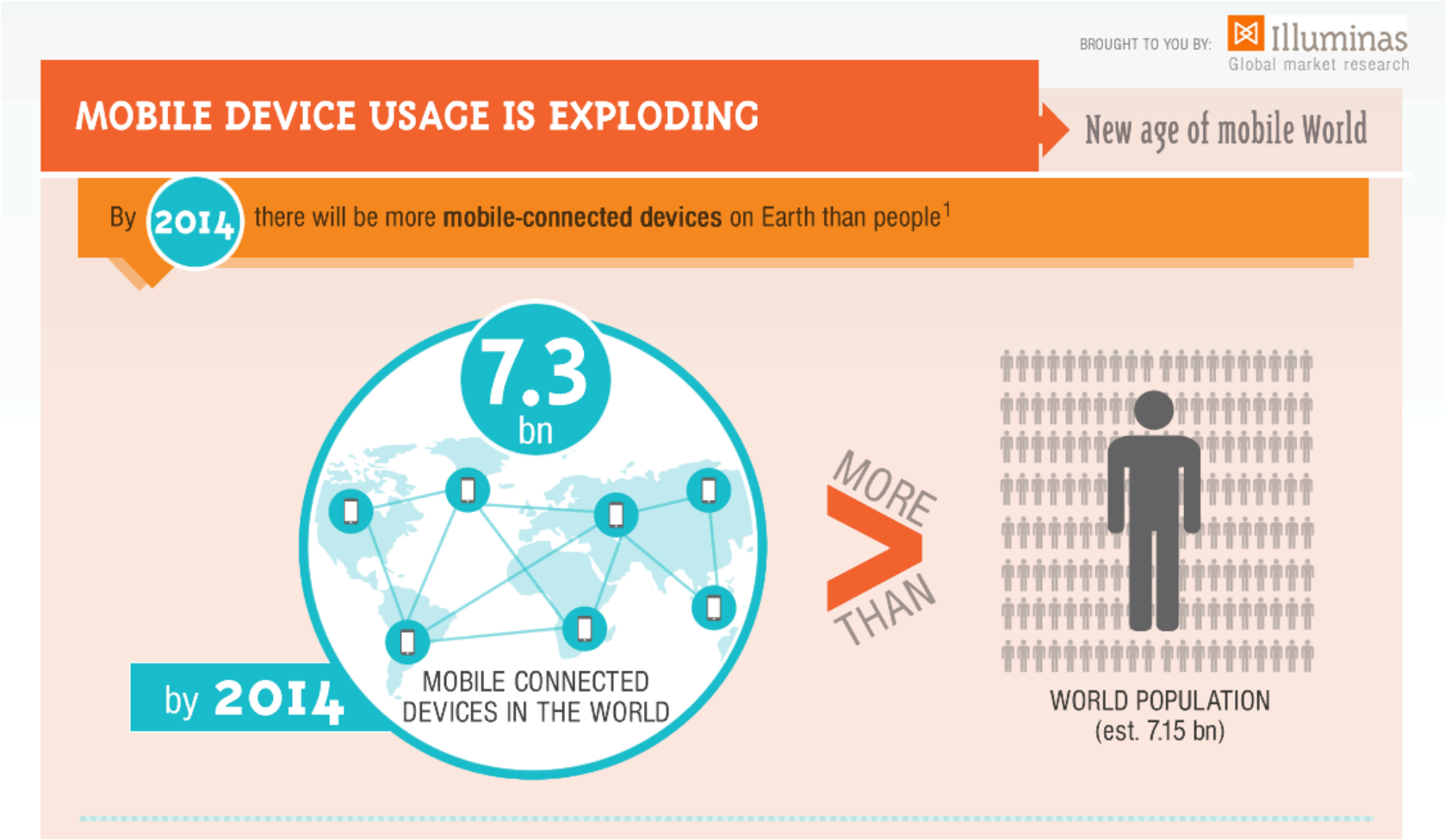
Sept. 10, 2013

QUALCOMM®

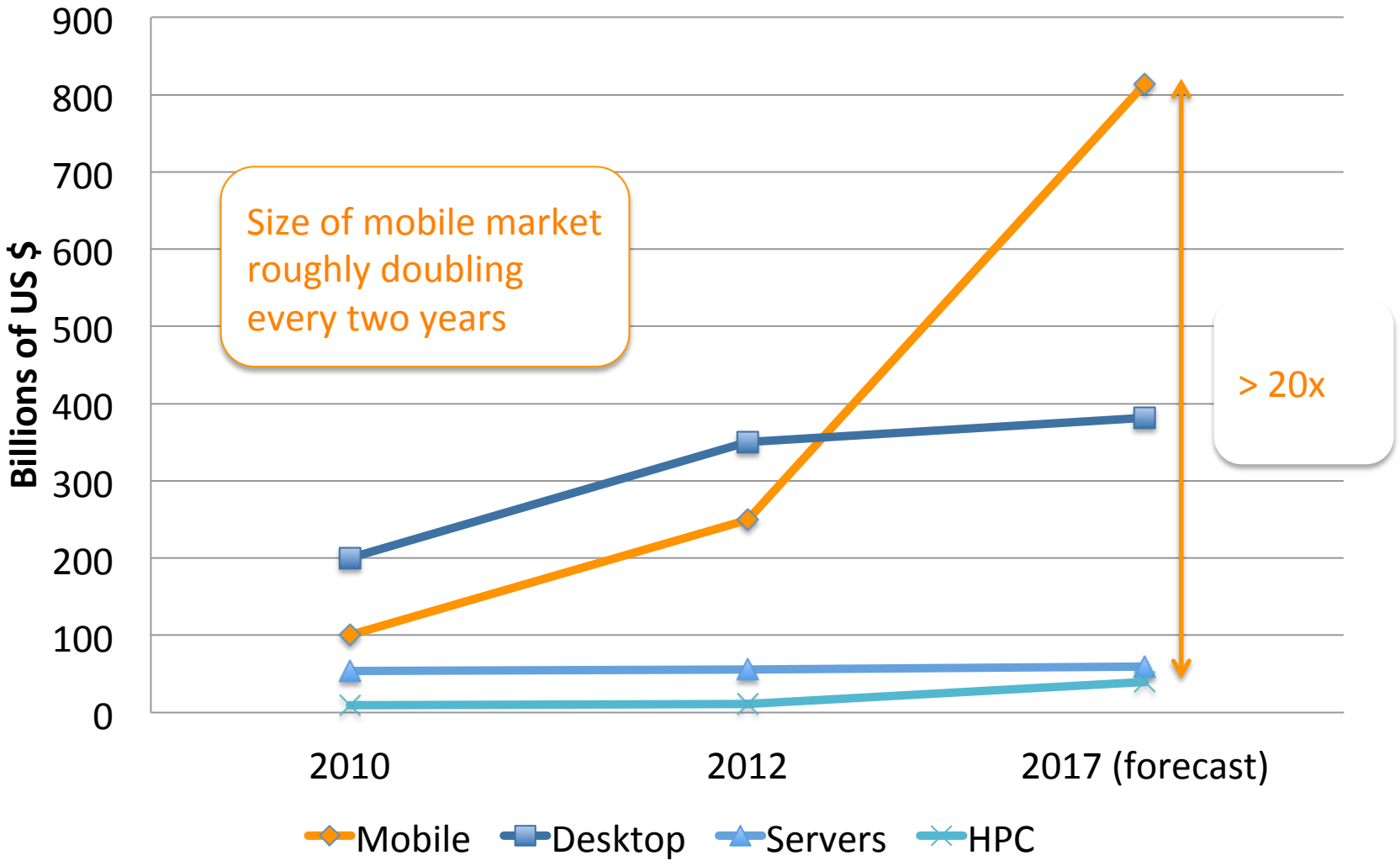
Outline

- Mobile computing market -- what do users care about?
- Mobile computing landscape -- how does the platform look like?
- Applications and developers -- who is writing code for mobile?
- Browser wars revisited -- why does it matter?
- Parallel heterogeneous programming made easier using MARE

Mobile Market

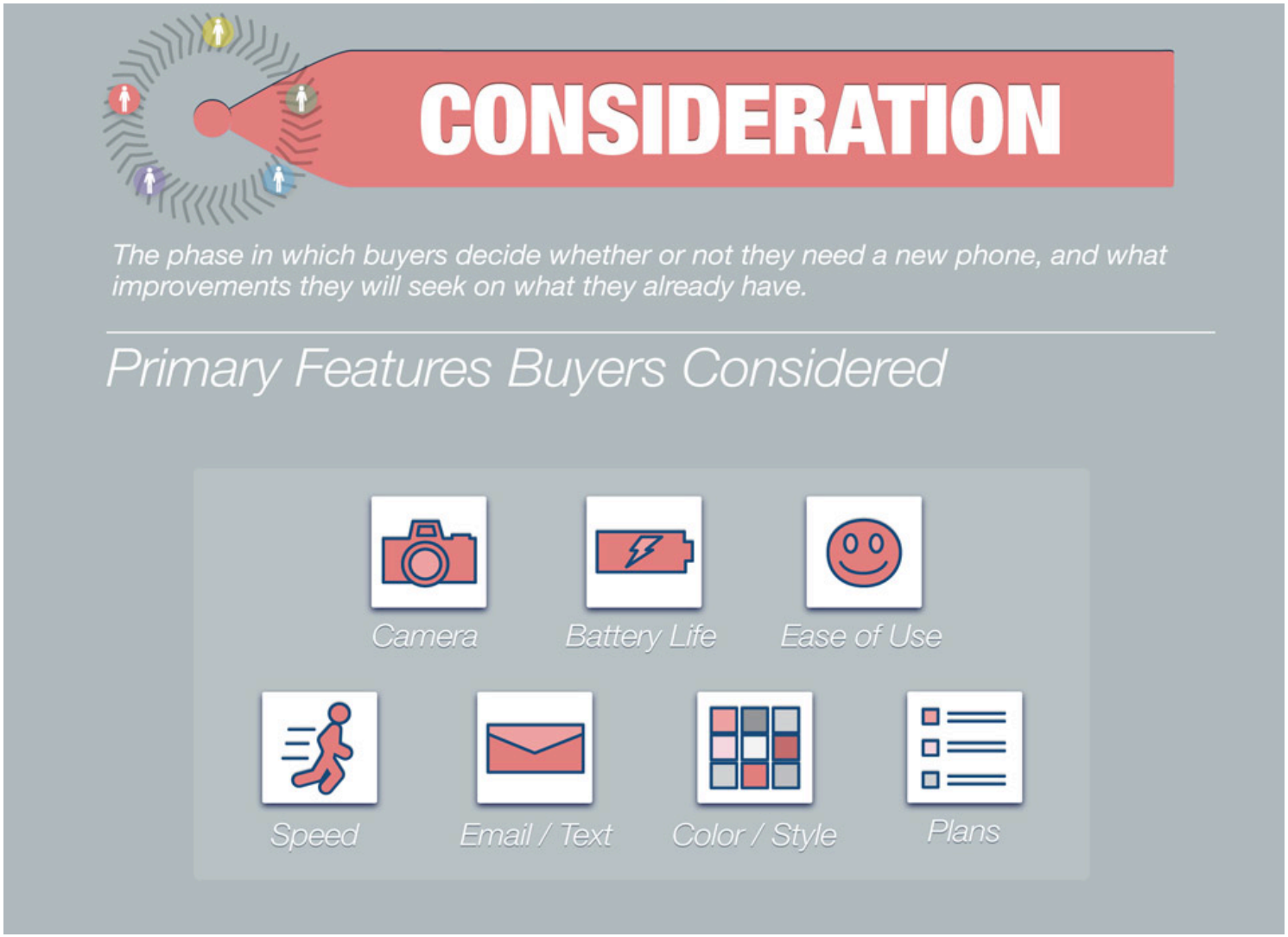


Mobile Market Size



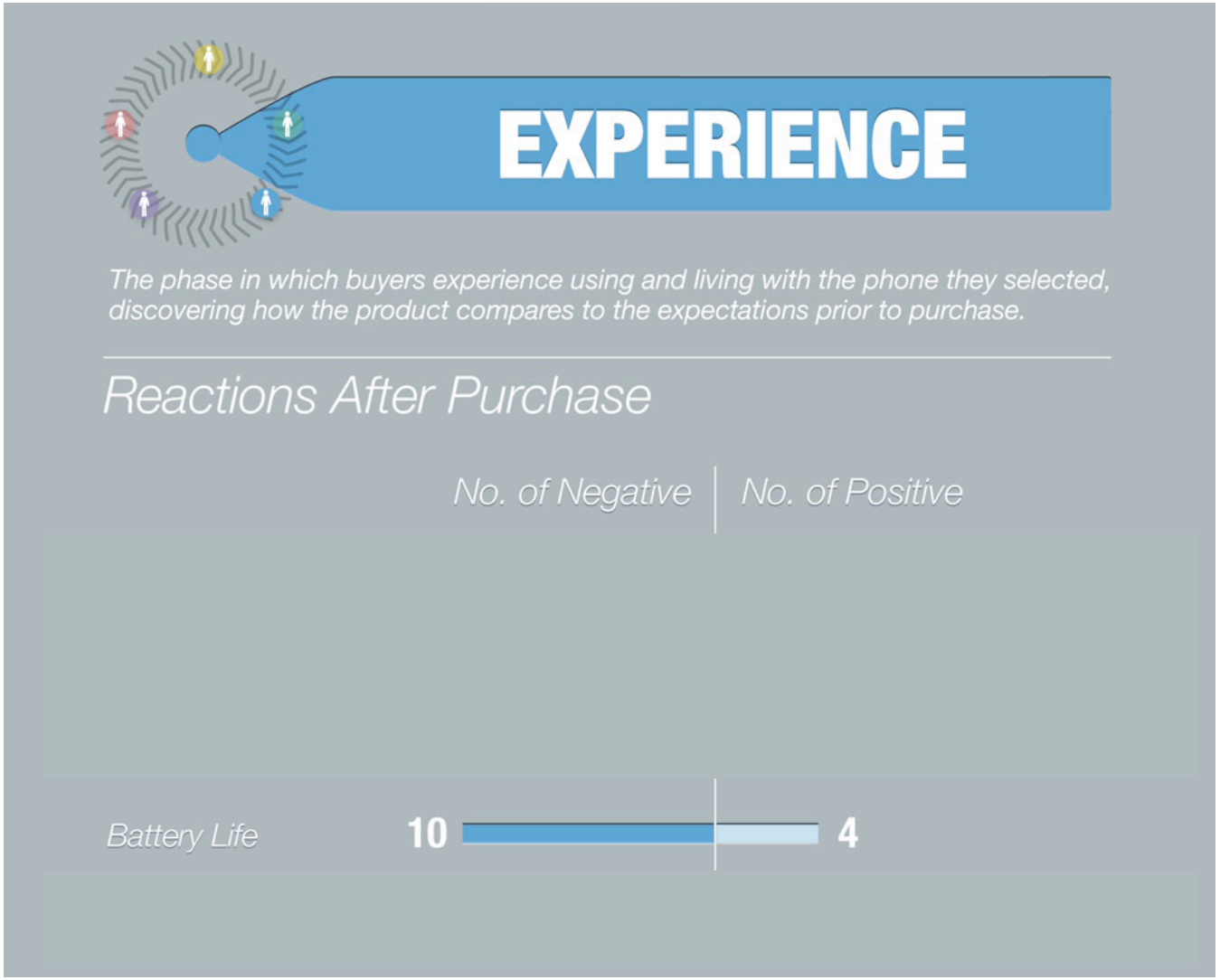
Source: IDC. HPC forecast Intersect360

What do users care about?



McKinsey's Consumer and Shopping Insights

What about next morning?



McKinsey's Consumer and Shopping Insights

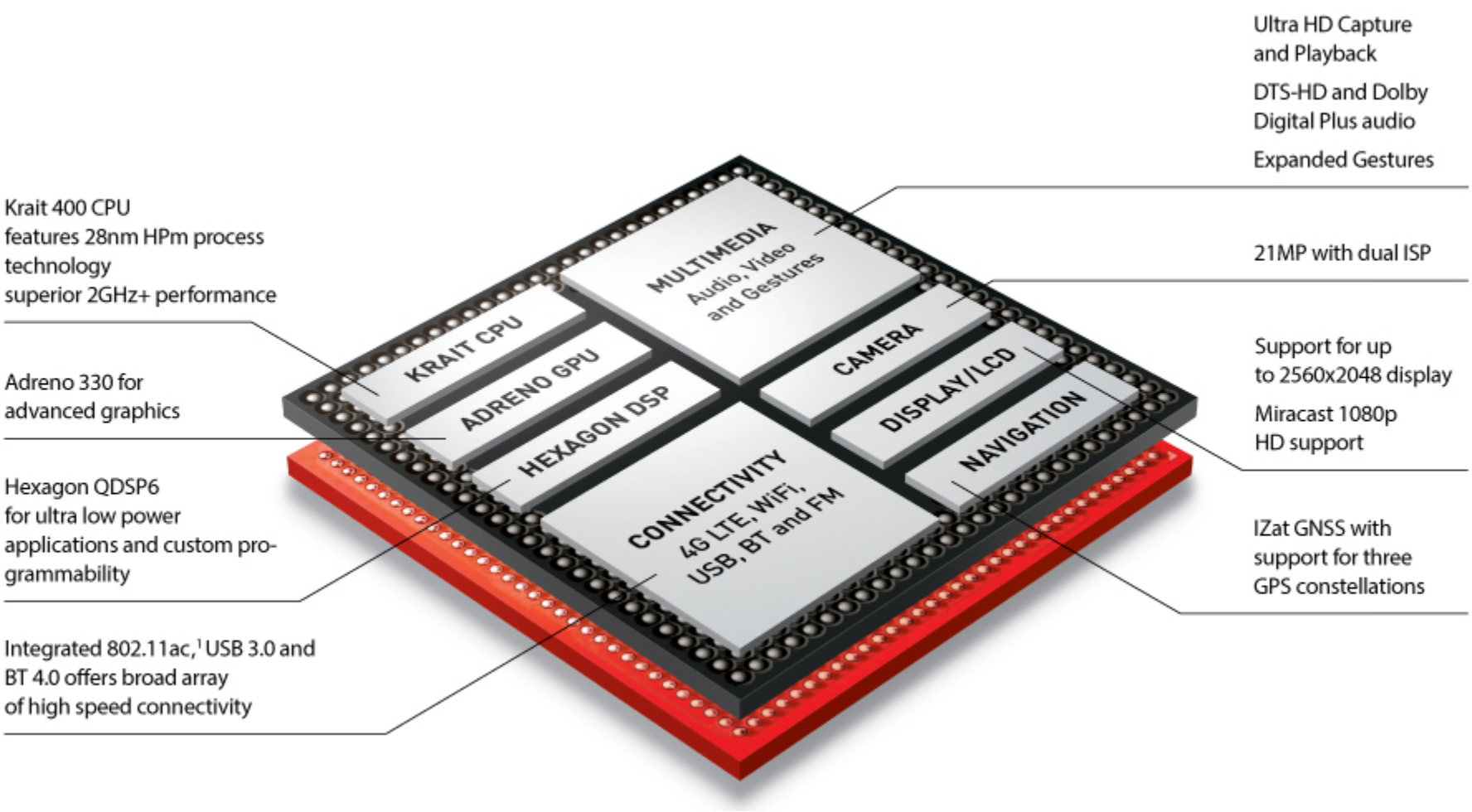
The Internet of Everything

Connectivity	Context	Control
An estimated 25B devices connected by 2020, talking to each other, making choices, and taking decisions	Use context awareness to filter the data that reaches every person: time, location, taste	Allow users to control and customize experience through new, mobile centric interfaces

Enabling whole new classes of applications, where power is even more critical

What’s in a smartphone?

Anatomy of a Mobile SoC: Qualcomm Snapdragon 800



Source: http://www.qualcomm.com/snapdragon/processors/800

Mobile Constrains

Energy

Thermal

Mobile Applications

Connectivity

Social networking, sharing:
Media rich, data heavy

Gaming

On device and multiplayer games with realistic graphics

Browsing

Information gateway

Imaging

Computational photography, search and processing of images and video

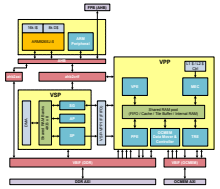
Mobile Software Development

Developers

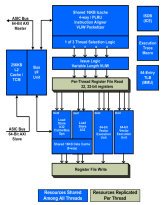
- Think in terms of services and “features”
- Portable apps, performance not the main concern
- Use JavaScript frameworks

Gap!

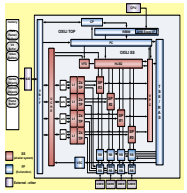
Hardware Accel.



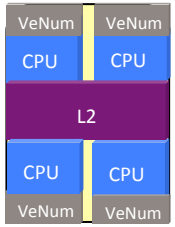
DSP



GPU



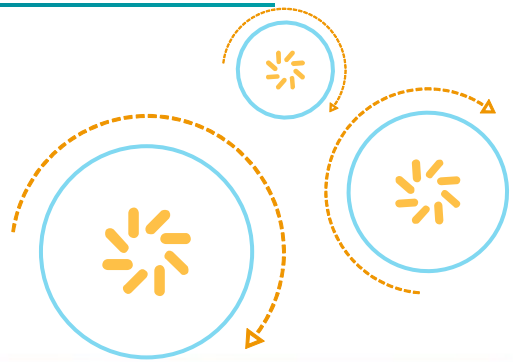
CPU(s)



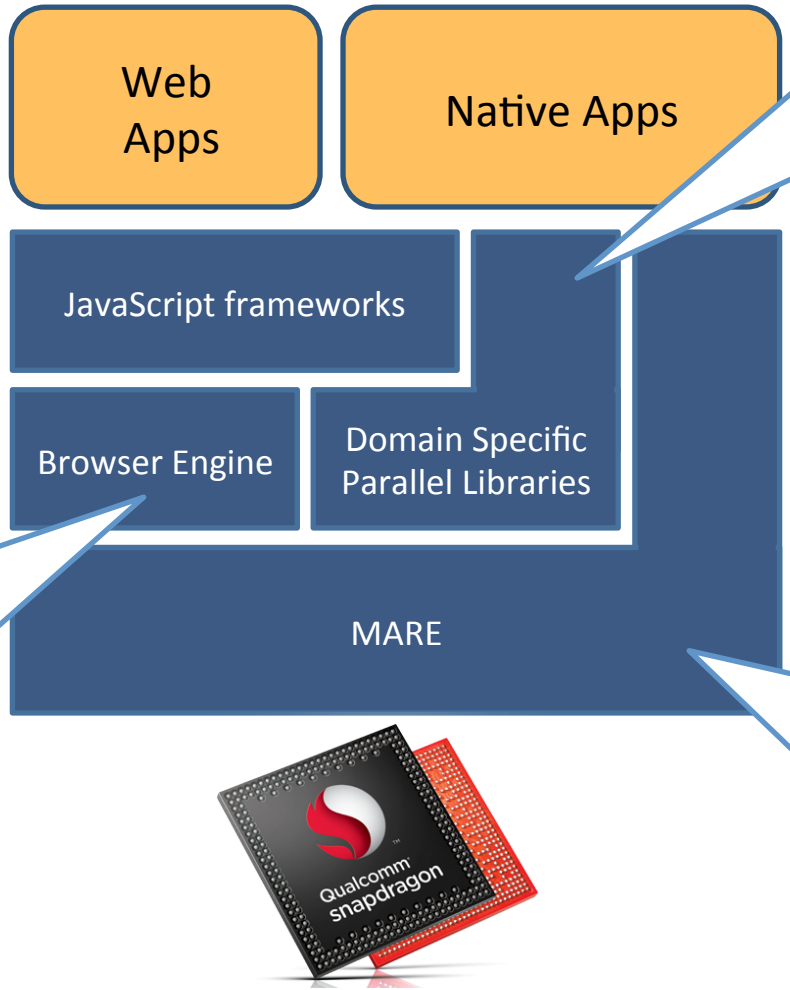
High Power Efficiency

Low Power Efficiency

How can we easily write parallel applications that use all available cores in a battery-powered device?



Our Vision of a Mobile Software Stack



Performance: Domain Specific libraries
- Exploit domain knowledge to provide composable libraries for all programmers
- Hide hardware complexity

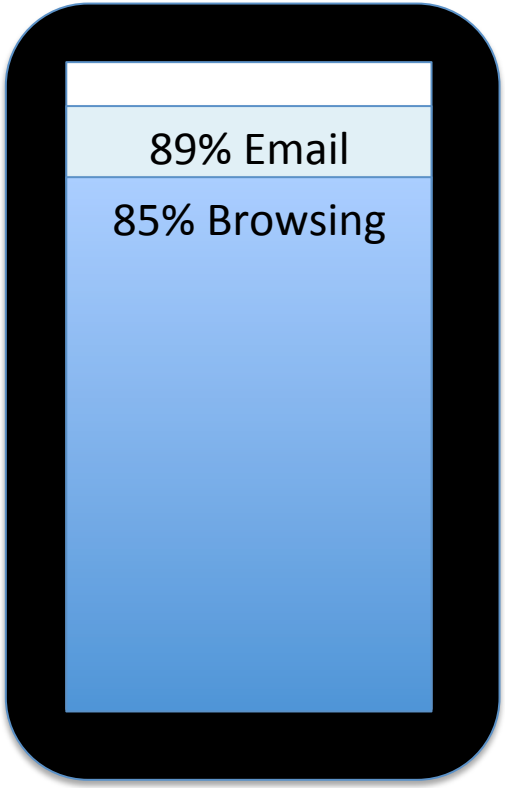
Portability: Zoomm Web App Engine
- Use of concurrency to optimize execution of Web Apps
- Hardware exploitation through the browser

Programmability: MARE
- Parallel, heterogeneous programming made easier
- Power and performance optimizations

Browsers: why do we care?

Web Browser Usage

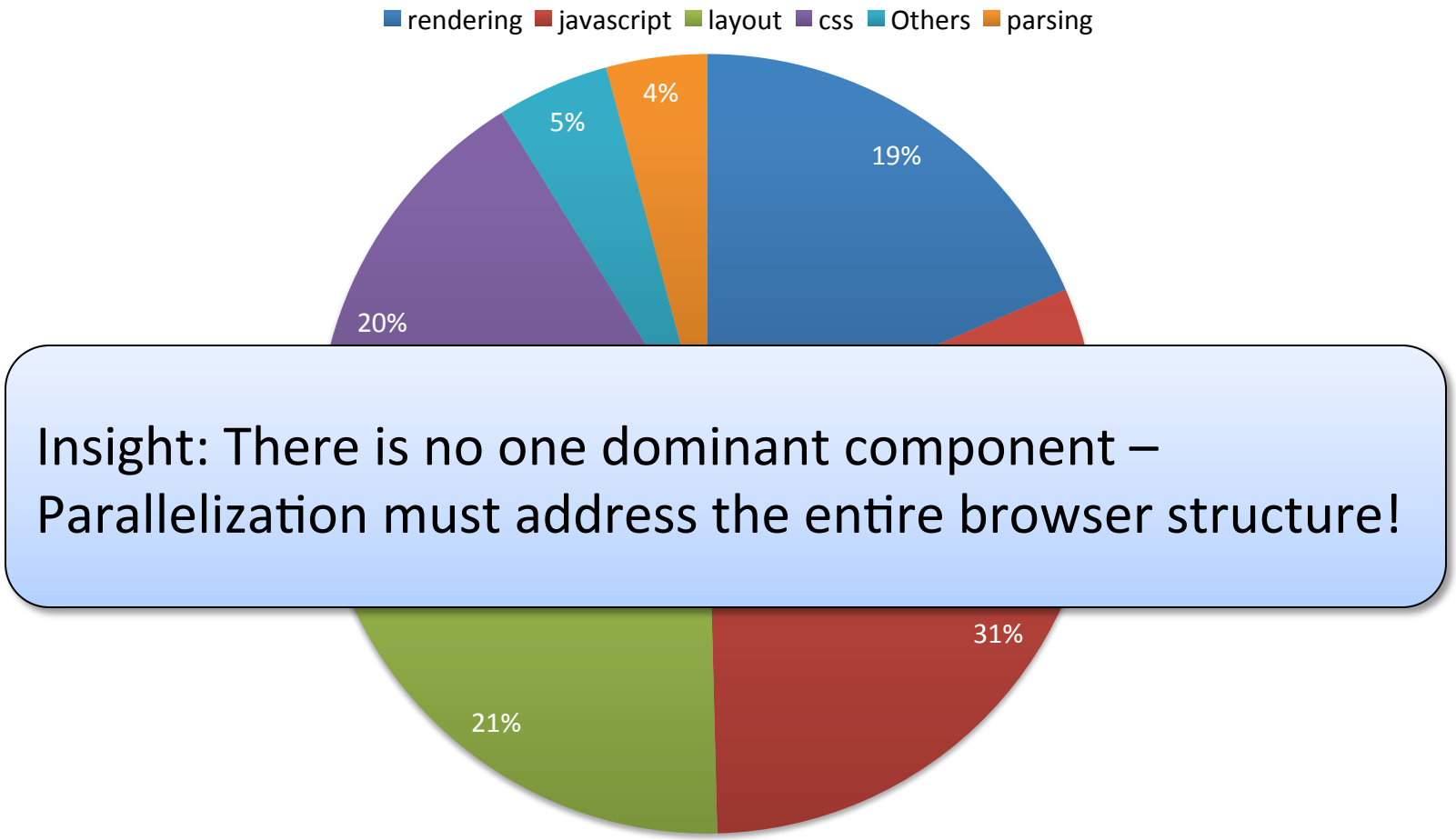
Information access



Web Applications

Source: Illuminas, 2013

Browser Execution Time Breakdown

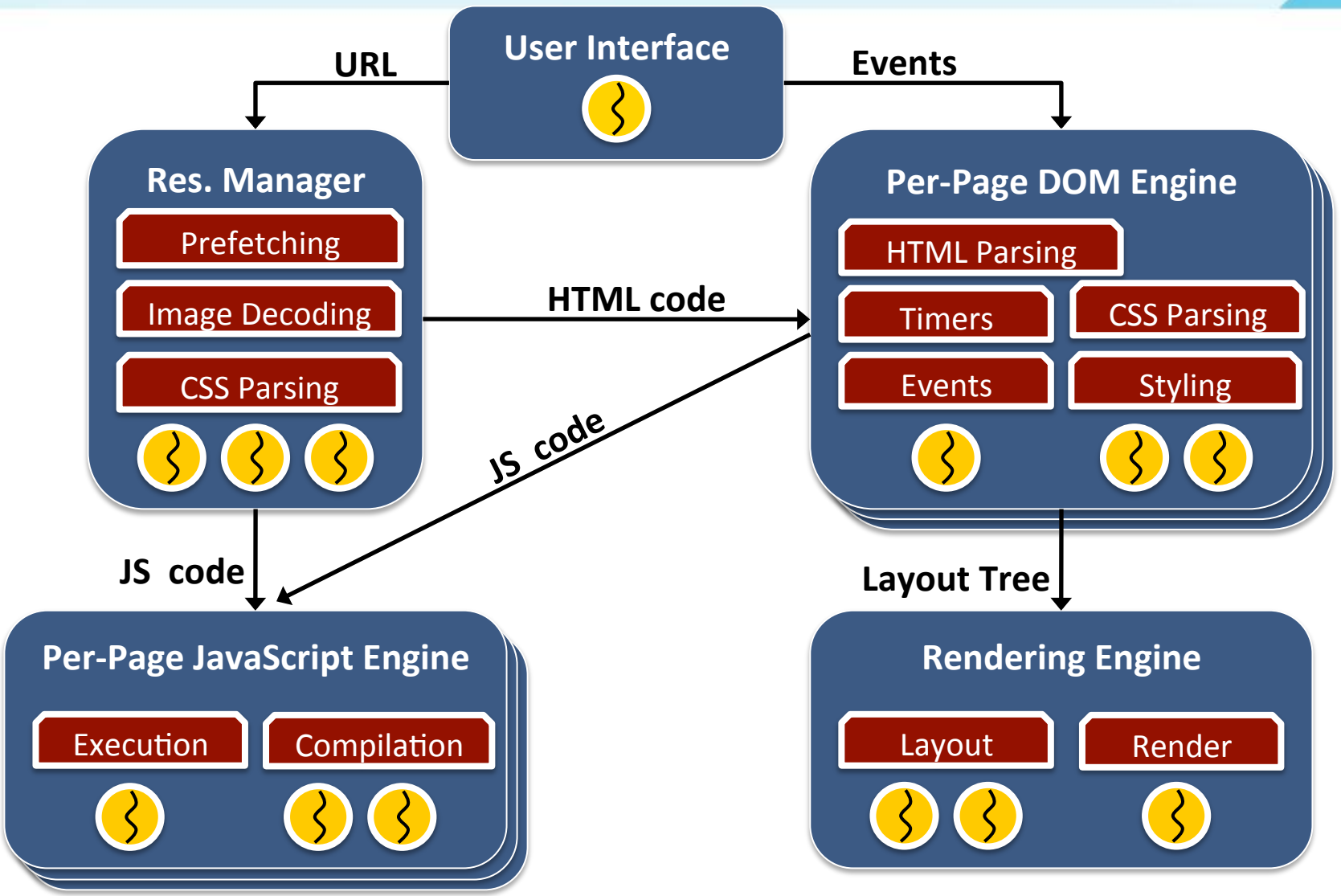


Insight: There is no one dominant component – Parallelization must address the entire browser structure!

ARM Cortex A9 Execution Time

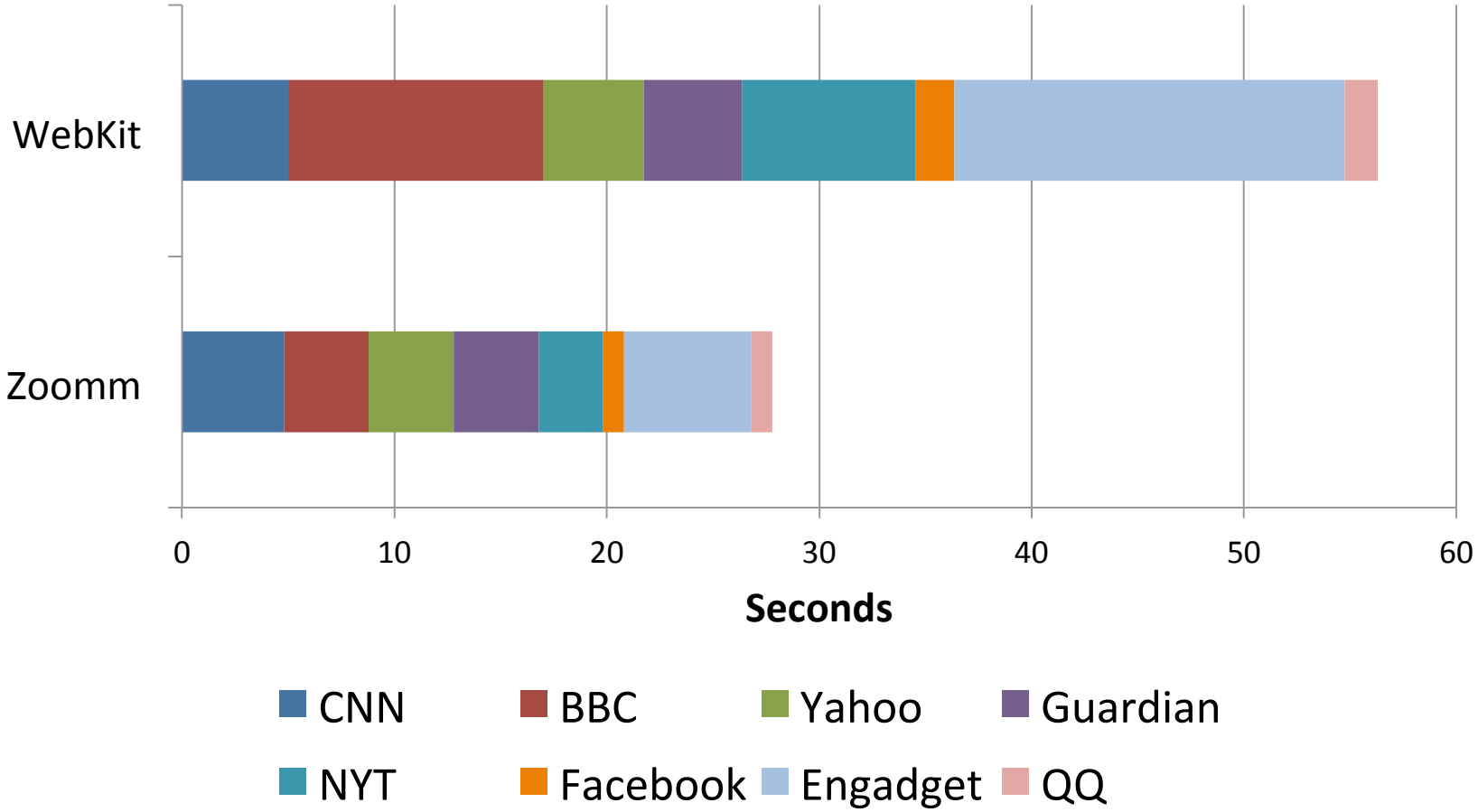
Average over the top 30 Alexa sites (May 2010), WebKit browser on a 400MHz Cortex A9, Linux

Zoomm: Pervasive Concurrency



Cascaval et al.: Zoomm: a parallel web browser engine for multicore mobile devices, PPoPP 2013

Zoomm: Page Load Performance



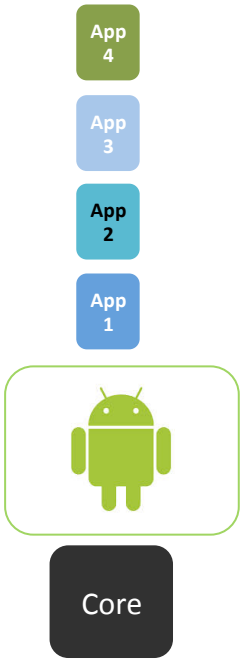
HTC Jetstream, MSM8660, 2-core, 1.5GHz, Aug 2012

Qualcomm MARE

Multicore Asynchronous Runtime Environment

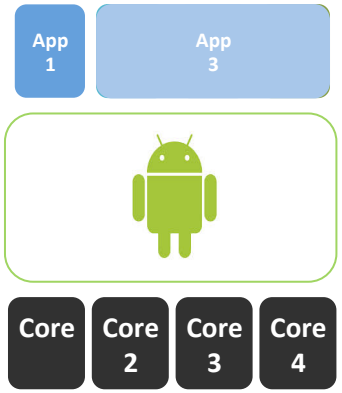
The Smartphone of 2010

- All applications run on a single-core Qualcomm Snapdragon processor at 1GHz



The Smartphone of 2013

- Multiple applications can simultaneously run on a quad-core Snapdragon at 2+GHz



What is Qualcomm MARE?

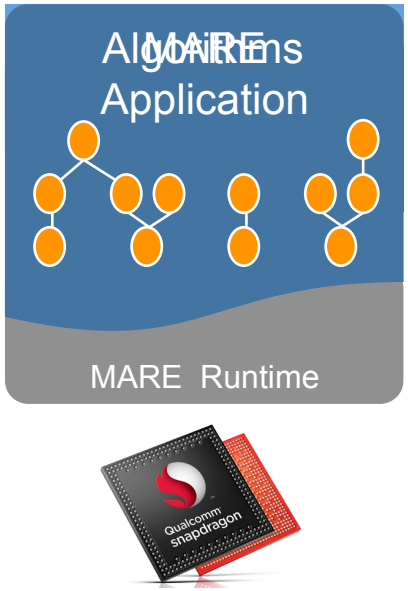
- MARE is a programming model and a runtime system that provides simple yet powerful abstractions for parallel, power-efficient software
  - Simple C++ API allows developers to express concurrency
  - User-level library that runs on any Android device, and on Linux, Mac OS X, and Windows platforms

- The goal of MARE is to reduce the effort required to write apps that fully utilize heterogeneous SoCs

MARE Workflow

Focus on your application and not on the hardware

- Understand algorithms
- Partition algorithms into independent units of work
- Setup task dependencies
- Link with MARE Runtime



Qualcomm MARE API Concepts

- **Tasks** are units of work that can be asynchronously executed
- **Groups** are sets of tasks that can be canceled or waited on

Advantages of using Qualcomm MARE

Simple	Productive	Efficient
Tasks are a natural way to express parallelism	Focus on application logic, not on thread management	Task dependences allow the MARE runtime to perform more intelligent scheduling decisions

Hello World!

```
#include <stdio.h>
#include <mare/mare.h>

int main() {
  mare::runtime::init();                                          // Initialize MARE

  auto hello = mare::create_task([] {printf("Hello ");});         // Create task
  auto world = mare::create_task([] {printf("World!");});         // Create task

  hello >> world;                                                 // Set dependency

  mare::launch(hello);                                            // Launch hello task
  mare::launch(world);                                            // Launch world task

  mare::wait_for(world);                                          // Wait to complete

  runtime::shutdown();                                            // Shutdown MARE
  return 0;
}
```

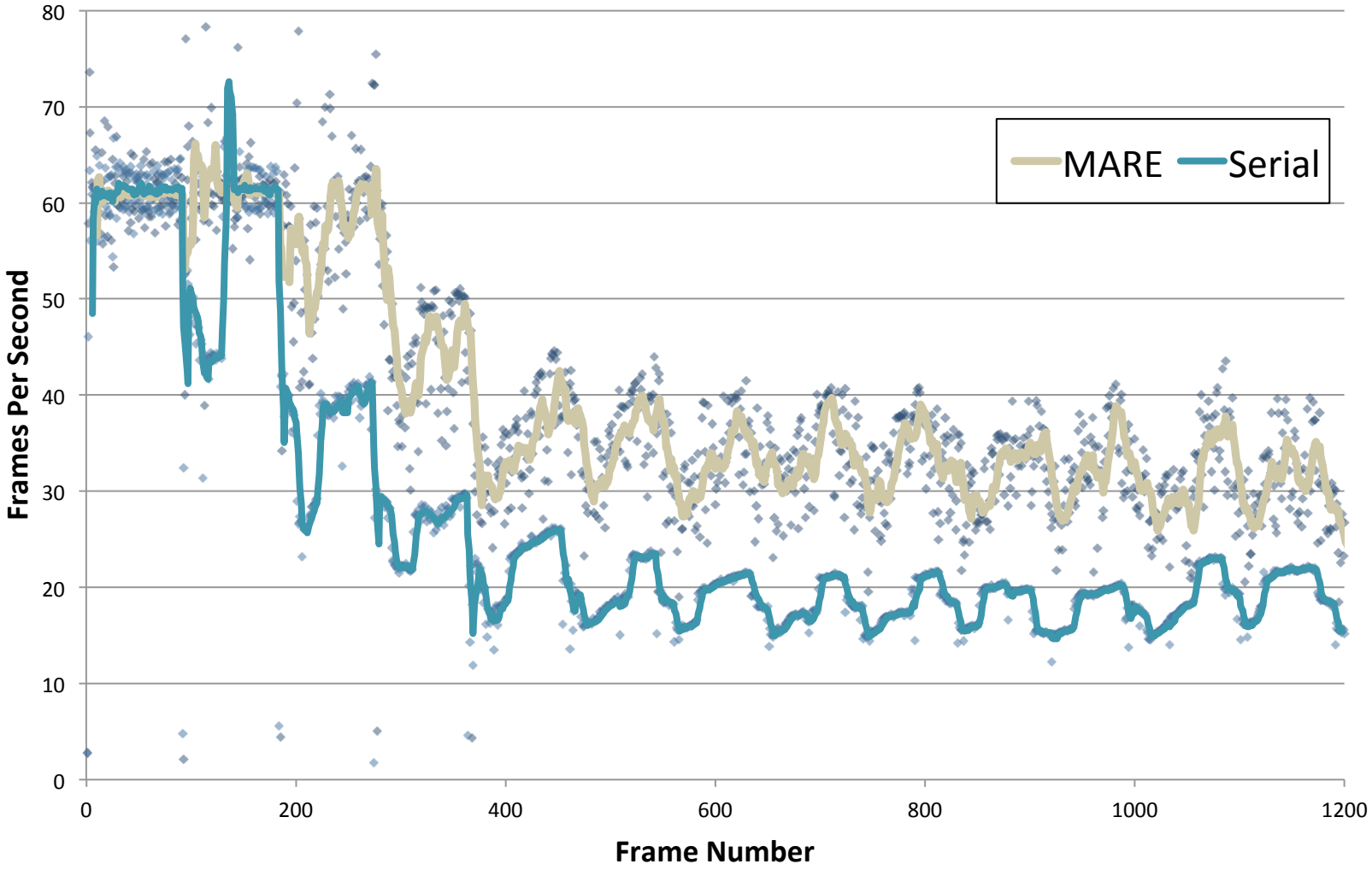
Parallel Programming Patterns in MARE

- MARE offers several commonly used parallel patterns:
  - `mare::pfor_each` processes the elements of a collection
  - `mare::pscan` performs an in-place parallel prefix operation for the elements of a collection.
  - `mare::transform` performs a map operation on the elements of a collection
- Programmers can also create their own patterns using the MARE API
- Parallel data structures: concurrent queues, hash tables, etc.

Bullet Physics Parallelization using MARE

- Three major components: about 75% of total execution time
- Constraint Solver
  - Finds non-interacting groups of objects and solves as independent tasks
  - Coarse-grained task-parallelization
- Rendering
  - Very coarse-grained task-parallelization: offload to a separate thread to allow for continuous simulation
- Narrow Phase Collision Detection
  - Fine-grained data-parallelization
- About 2x FPS improvement on typical mobile devices
  - Dramatically improves the smoothness of rendering and overall user experience

Bullet Physics Parallelization



How can You try MARE?

- Stay tuned!
- Planning preview availability on Qualcomm Developer Network
  - Fall 2013 Beta release for the multicore version
- We Want Your Feedback!
  - What scenarios will you use it for?
  - Usability and features of the library

Acknowledgments

- Manticore team
  - Pablo Montesinos Ortego, Michael Weber, Wayne Piekarski, Behnam Robatmili, Dario Suarez-Gracia, Vrajesh Bhavsar, Jimi Xenidis, Han Zhao, Kishore Puskuri
- Interns
  - Madhukar Kedlaya, Christian Delozier, Freark Van Der Berg, Christoph Kershbaumer
- Former members
  - Mehrdad Reshadi, Seth Fowler, Alex Shye, Dilma DaSilva
- Qualcomm
  - Nayeem Islam, Mark Bapst, Charles Bergan, Matt Grob, Ben Gaster
- MulticoreWare
  - Wen-Mei Hwu, Mark Allender, Kevin Wu

Challenges

Power and performance

Custom hardware for power efficiency. Heterogeneity is the game.

Expertly designed and optimized frameworks that hide hardware complexity

Programmability

Tools and languages to enable programmers to understand and express concurrency and application semantics

Composable building blocks

Mobile has exciting applications and a huge impact

Legal Disclaimers

All opinions expressed in this presentation are mine and may not necessarily reflect those of my employer.

Qualcomm and Snapdragon are trademarks of Qualcomm Incorporated, registered in the United States and other countries. All Qualcomm Incorporated trademarks are used with permission.

Other products and brand names may be trademarks or registered trademarks of their respective owners.