Parallel Programming for the Web

Stephan Herhut, Richard L. Hudson, Tatiana Shpeisman, Jaswanth Sreeram
HotPar ’12, June, 7th 2012
JavaScript* – What You Need To Know

• It is not Java*

• Blend of many programming paradigms
  • Object oriented with prototypes
  • Higher-order functions and first class function objects
  • Dynamically typed and interpreted

• Safety and security built in
  • Requirement for web programming
  • Managed runtime
  • No pointers, no overflows, …

• Designed for portability
  • Fully abstracts hardware capabilities
Concurrency in JavaScript*

- Cooperative multi-tasking
  - Scripts compete with the browser for computing resources
  - Event driven execution model
- Concurrent programming mindset
  - Asynchronous call-backs for latency hiding
- Fully deterministic
  - Run-to-completion semantics
  - No concurrent side effects
- **No support for concurrent execution**
  - Single threaded evaluation of JavaScript
Yet Another Parallel Programming API?
Design Considerations
Language Design with the Web in Mind

1. Ease of use
   - Build on developer’s existing knowledge
   - Allow for mash-up of sequential and parallel code

“Meant to be a scripting language [...] for the designer, the amateur programmer, the beginner programmer”

Brendan Eich, CTO Mozilla
Language Design with the Web in Mind

1. Ease of use
   - Build on developer’s existing knowledge
   - Allow for mash-up of sequential and parallel code

2. Platform independent
   - Support all kinds of platforms, parallel or not
   - Perform well on different parallel architectures (multi-core, GPUs, ...)

3. Suitable for the Open Web
   - Meet existing safety and security promises
   - Needs to be reasonably easy to implement in JavaScript JIT engines

Challenge: meet these criteria and get good performance
Design Choices

• Performance portability
  ⇒ Use High-Level Parallel Patterns

• Deterministic execution model
  ⇒ No side effects: shared state is immutable
  ⇒ Require commutative and associative operators
  ⇒ No magic: floating point anomalies may still occur

• Support mash-up coding
  ⇒ All code still written purely in JavaScript
  ⇒ Looks like JavaScript*, behaves like JavaScript*

• Maintain JavaScript*’s Safety and Security
  ⇒ Use fully managed runtime
River Trail API
Three Pillar Approach

- Data structure: `ParallelArray`
  - Immutable, dense and homogeneous
- Six Methods: `map`, `combine`, `reduce`, `scan`, `filter`, `scatter`
  - Provide the basic skeletons for parallel computing
  - Typically creates a freshly minted `ParallelArray`
- Elemental functions (kernel functions)
  - Written purely in JavaScript
  - Side effect free

```javascript
pa = new ParallelArray([1, 2, 3, 4]);

pa.map(function (v) { return v+1; })
```
An Example: Grayscale Conversion

```javascript
pixelData.map(toGrayScale)
  .map(function toRGBA(color) {
    return [color, color, color, 255];
  })
```
Prototype Implementation
Compiling River Trail (Prototype)

- Type inference
  - Infers array types and shapes
  - Checks for side effects
- Representation analysis
  - Computes bounds on local variables
  - Updates type information of known Integer numbers
- Static memory allocation
- Bounds check elimination
- Code generation
  - Emits OpenCL code
Compiling River Trail (Prototype)
Performance Results: Particle Physics

Particle model \((O(n^2))\) computed using River Trail on a 2\textsuperscript{nd} Generation Core i7 with 4 cores

http://github.com/RiverTrail/RiverTrail/wiki
Performance Results: Matrix Matrix Multiply

$O(n^3)$ dense matrix matrix multiplication on 1000 x 1000 element matrices; dual-core 2nd Generation Core i5 with HyperThreading enabled and 4GB RAM; JavaScript* benchmarks use Firefox 8
Status Quo

• Open source Firefox prototype available on GitHub
  • Pre-built binary extension for Firefox 12
  • Sequential library fall back for other browsers
• ECMAScript proposal of the full API published
  • Removes many limitations of the prototype
• First sequential implementation for SpiderMonkey
  • Lives in Mozilla’s IonMonkey branch
  • Intended as API testing vehicle

http://github.com/RiverTrail/RiverTrail/wiki