Rationale for Map-Reduce

Map-reduce idea: An Example (part 1)

This example uses JavaScript.

```
  // A trivial example:
    alert("I'd like some Spaghetti!");
    alert("I'd like some Chocolate Mousse!");
```

The above can be improved to the following code:

```
function SwedishChef( food )
{
    alert("I'd like some " + food + "!");
}

SwedishChef("Spaghetti");
SwedishChef("Chocolate Mousse");
```

Map-reduce idea: An Example (part 2)

```
    alert("get the lobster");
    PutInPot("lobster");
    PutInPot("water");

    alert("get the chicken");
    BoomBoom("chicken");
    BoomBoom("coconut");
```

The above can be improved to the following code using function pointers!

```
function Cook( i1, i2, f )
{
    alert("get the " + i1);
    f(i1);
    f(i2);
}

Cook( "lobster", "water", PutInPot );
Cook( "chicken", "coconut", BoomBoom );
```

Map-reduce idea: An Example (part 3)

Functions can be anonymous (like classes)

```
Cook( "lobster",
      "water",
      function(x) { alert("pot " + x); }  );
Cook( "chicken",
      "coconut",
      function(x) { alert("boom " + x); } );
```

Map-reduce idea: An Example (part 4)

```
var a = [1,2,3];
for (i=0; i<a.length; i++) {
    a[i] = a[i] * 2;
}
for (i=0; i<a.length; i++) {
    alert(a[i]);
}
```

Doing something to every element of an array can be done by a function using a function pointer for what needs to be done.

```
function map(fn, a)
{
    for (i = 0; i < a.length; i++) {
        a[i] = fn(a[i]);
    }
}

map( function(x){return x*2;}, a );
map( alert, a );
```

Map-reduce idea: An Example (part 5)

```
function sum(a)
{
    var s = 0;
    for (i = 0; i < a.length; i++)
        s += a[i];
    return s;
}

function join(a)
{
    var s = "";
    for (i = 0; i < a.length; i++)
        s += a[i];
    return s;
}

alert(sum([1,2,3]));
alert(join(["a","b","c"]));
```

Map-reduce idea: An Example (part 6)

```
function reduce(fn, a, init)
{
    var s = init;
    for (i = 0; i < a.length; i++)
        s = fn( s, a[i] );
    return s;
}

function sum(a)
{
    return reduce( function(a, b){ return a + b; }, a, 0 );
}

function join(a)
{
    return reduce( function(a, b){ return a + b; }, a, "" );
}
```

Map-reduce idea

- The `map` function is embarrasingly parallel.
- The `reduce` function is nearly embarrassingly parallel.
- Someone else can write generic map and reduce code that scales to thousands of systems with massive data sets that span thousands of disks and can tolerate failures of components.... How do we go around doing that?
- Now as long as we can recast our problem as a map-reduce problem, it will automatically scale to thousands of processes!

Map-reduce implementation

- How to scale a parallel program to thousands of processes successfully?
- How to deal with massive amounts of data that doesn't fit on one system?
  - Distributed file system.
  - Move code to data instead of the other way around.
- How to deal with machines and components failing while the program is running?

Reference

- Can Your Programming Language Do This? by Joel Spolsky.