Clojure Parallelism

- Basic set of parallel APIs:
- `pmap` `pvalues` `pcalls` (lazy, sequential, chunked)
- `reducers/fold` (work-stealing, fork-join)
- Custom with `future`, `agent`, etc.
- `core.async` pipelines (external lib)

Problems

- Powerful, but somewhat low level
- Not necessarily easy to use
- Even more to use correctly
- Inconsistencies with stateful xforms
- Chunk size dependency (`pmap`)

Can we approach it differently?

- Task oriented API
- Predictable semantic
- Easy to use
- Documented

The Parallel Library

- Experimenting ideas in a library
- https://github.com/reborg/parallel
- Documented, tested and benchmarked
- No dependencies (other than Clojure itself)

At a glance

- Modeled on existing functions from the stdlib
- Drop-in replacement (when possible)
- A few brand new functions
- Some specific transducers support

Current Line-up 1/4

Name	Description
`p/let`	Parallel `let` binding.
`p/slurp`	Parallel slurping files.
`p/count`	Transducer-aware parallel `core/count`.
`p/frequencies`	Parallel `core/frequencies`
`p/group-by`	Parallel `core/group-by`

Current Line-up 2/4

Name	Description
`p/update-vals`	Updates values in a map in parallel.
`p/external-sort`	Memory efficient, file-based, parallel merge-sort.
`p/sort`	Parallel `core/sort`.
`p/fold`	Transducer-aware `r/fold`.

Current Line-up 3/4

Name	Description
`p/min` and `p/max`	Parallel `core/min` and `core/max` functions.
`p/distinct`	Parallel version of `core/distinct`
`p/amap`	Parallel array transformation.
`p/armap`	Parallel array reversal with transformation.

Current Line-up 4/4

Name	Description
`xf/interleave`	Like `core/interleave`, transducer version.
`xf/pmap`	Like `core/pmap`, transducer version.
`xf/identity`	Alternative identity transducer to `core/identity`

DEMO TIME!