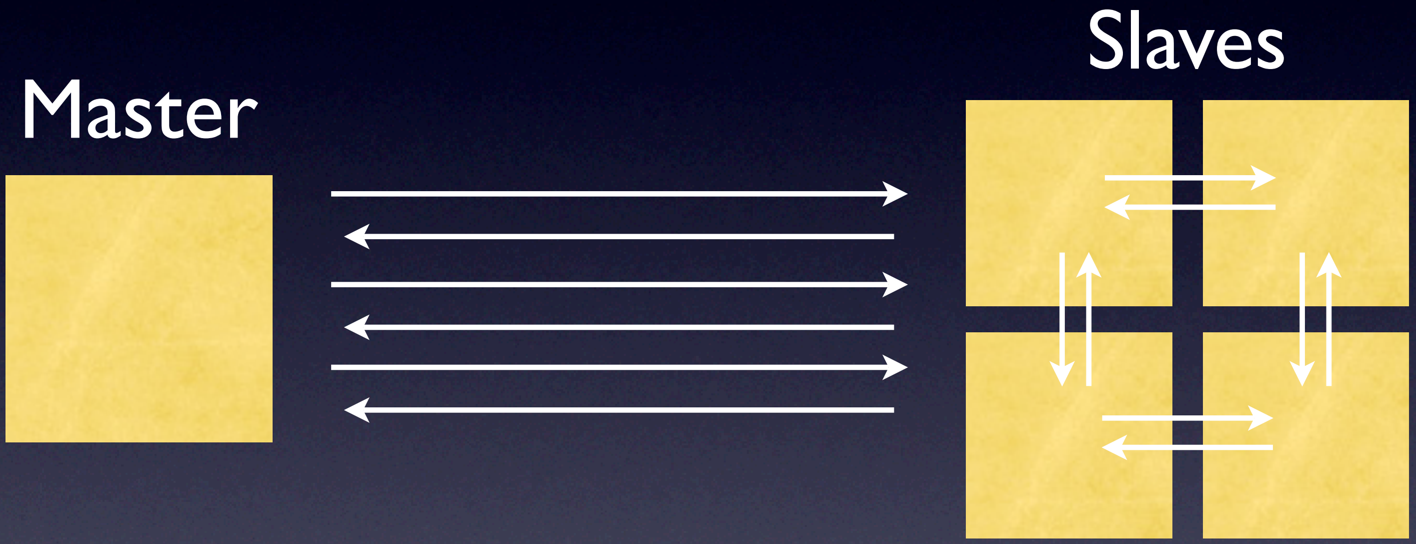
Loosely Dependent Parallel Processes

Complementary Paradigms

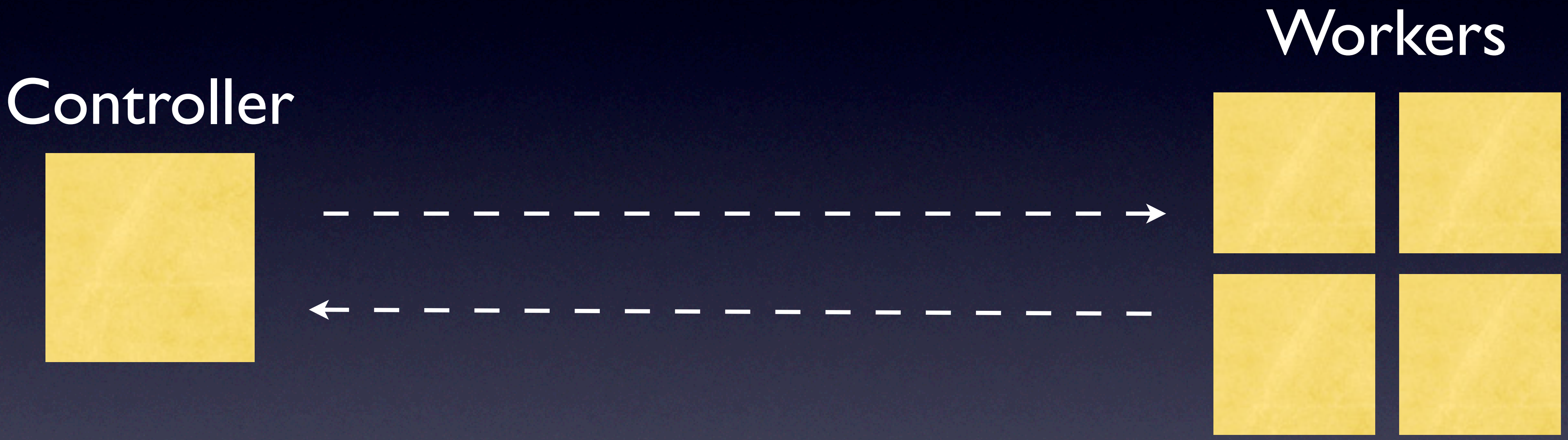
- Massively Parallel
- Task Farm

Massively Parallel



- MPI/shared memory

Task Farm



Controller

Workers

- Occasional network access
- E.g. BOINC

Integer Factorization

- Trial Division
- Quadratic Sieve
- Elliptic Curve Method

Trial Division

- Fast for small factors
- Necessary pre-processing for other methods

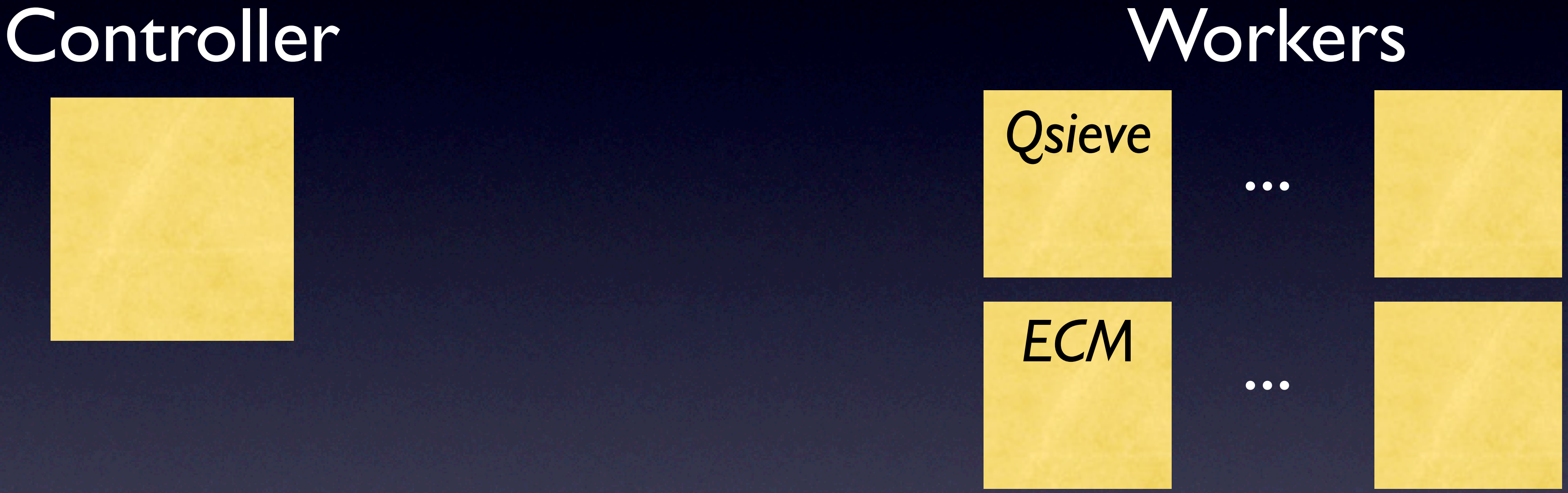
Quadratic Sieve

- Among the fastest (known) algorithms for “reasonably” sized primes
- Runtime $O\left(\exp\left(\sqrt{n \log \log \log n}\right)\right)$
- Relation discovering phase embarrassingly parallel

Elliptic Curve Method

- Probabilistic, embarrassingly parallel
- Runtime $O\left(\exp\left(\sqrt{p \log\log\log p}\right)\right)$
    - Dominated by size of *smallest factor*
- Use to peel off smaller factors

DSage implementation



Controller

Workers

Qsieve ...

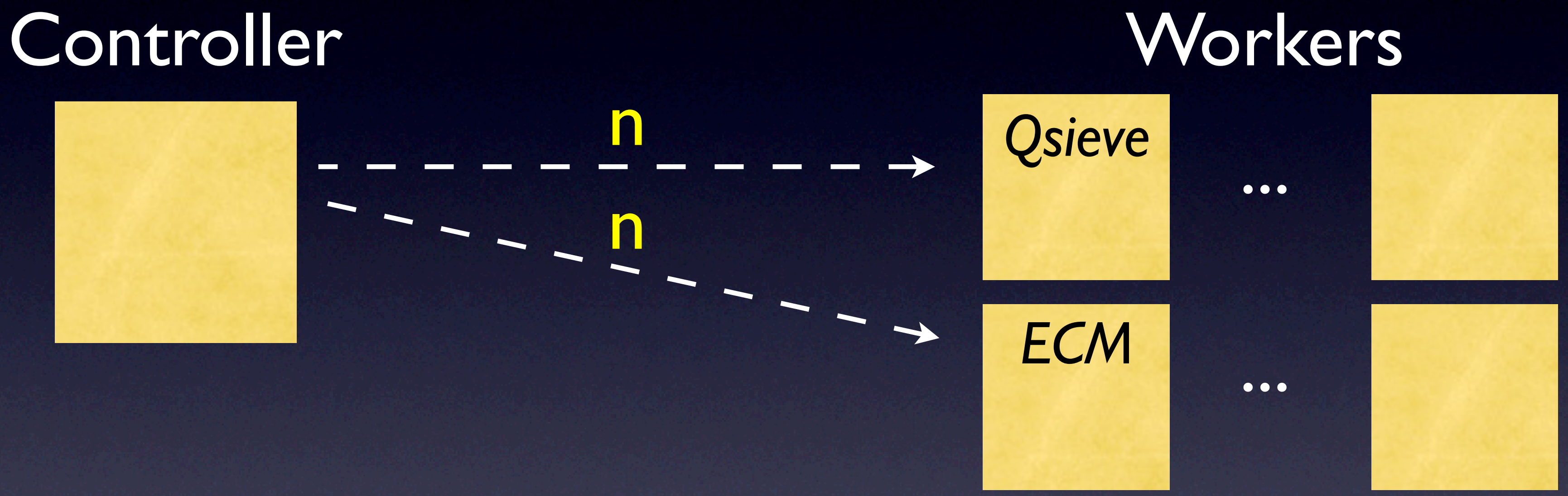
ECM ...

DSage implementation

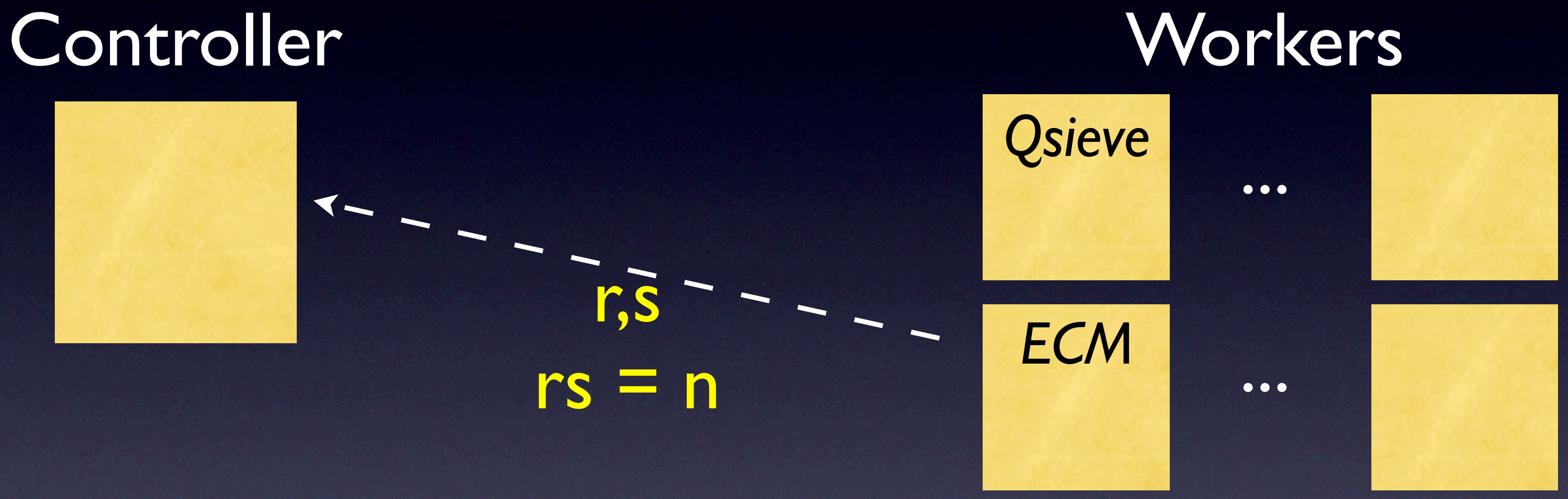
Controller

- factors = {n}
- While factors not all prime
    - Wait for factor r
    - Use GCD(-,r) to split factors
    - Start new ECM/Qsieve workers

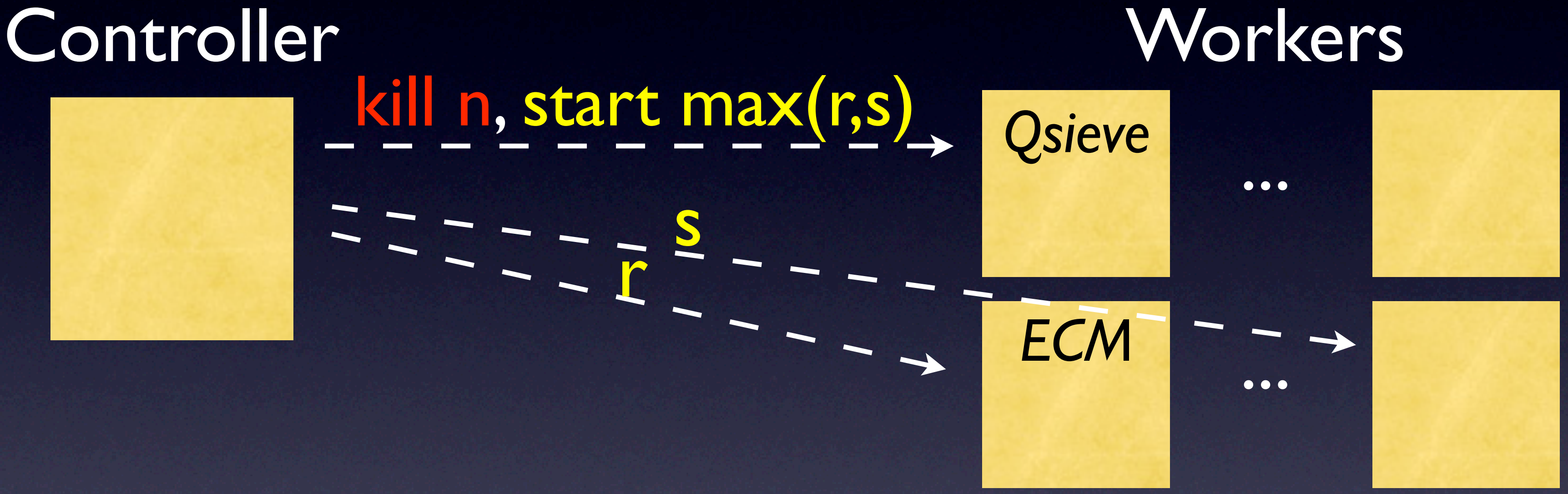
DSage implementation



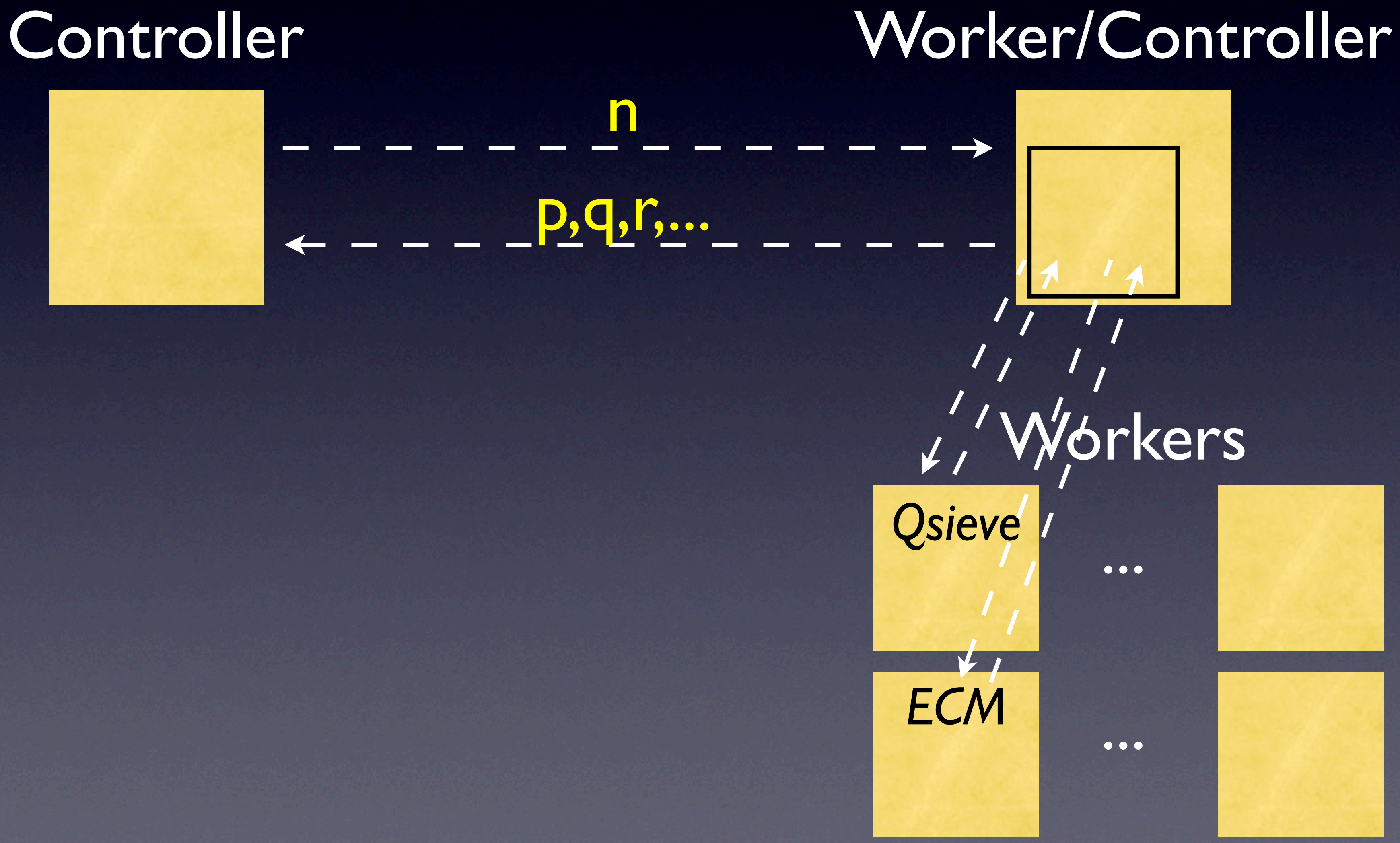
DSage implementation



DSage implementation



Offline Controllers



Communication Bottleneck

- All communication passes through server and client
- Currently extremely course-grained (workers listen only for kill)
- Obviously we can’t compete with MPI, but many almost-embarrassingly parallel problems don’t need that

Worker-to-Worker

- Pros
  - Can open up a much wider range of problems
    - E.g. periodically sharing boundary data
- Cons
  - Firewalls, etc.

Questions?