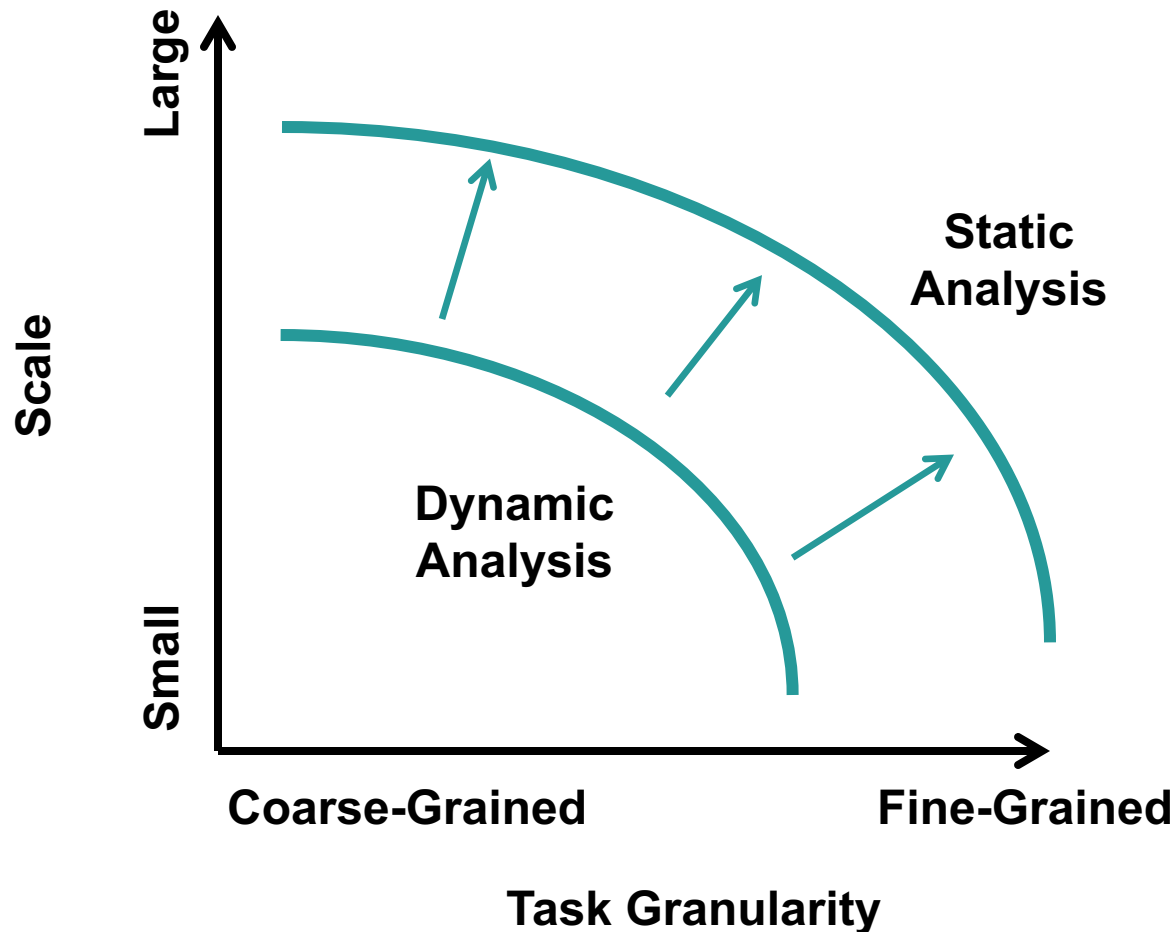




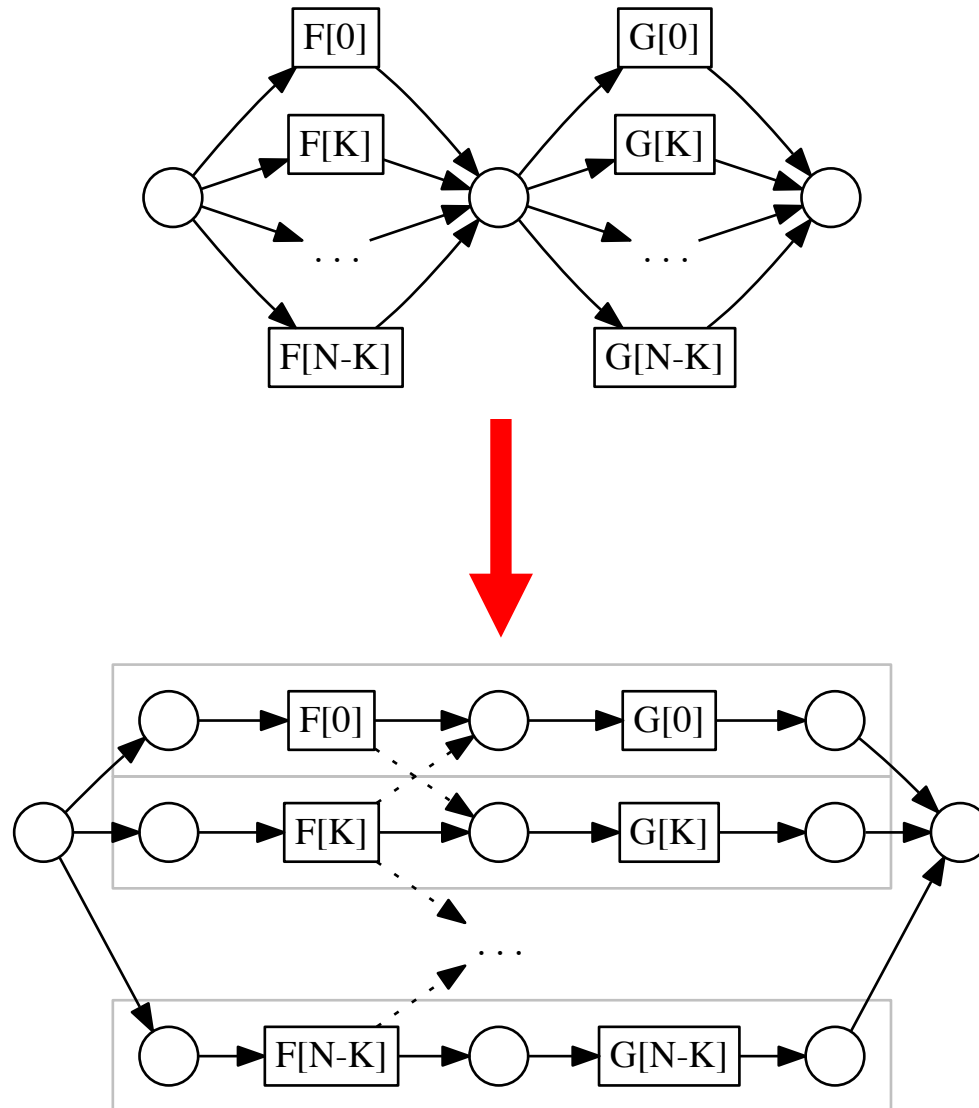
Regent Update

Elliott Slaughter

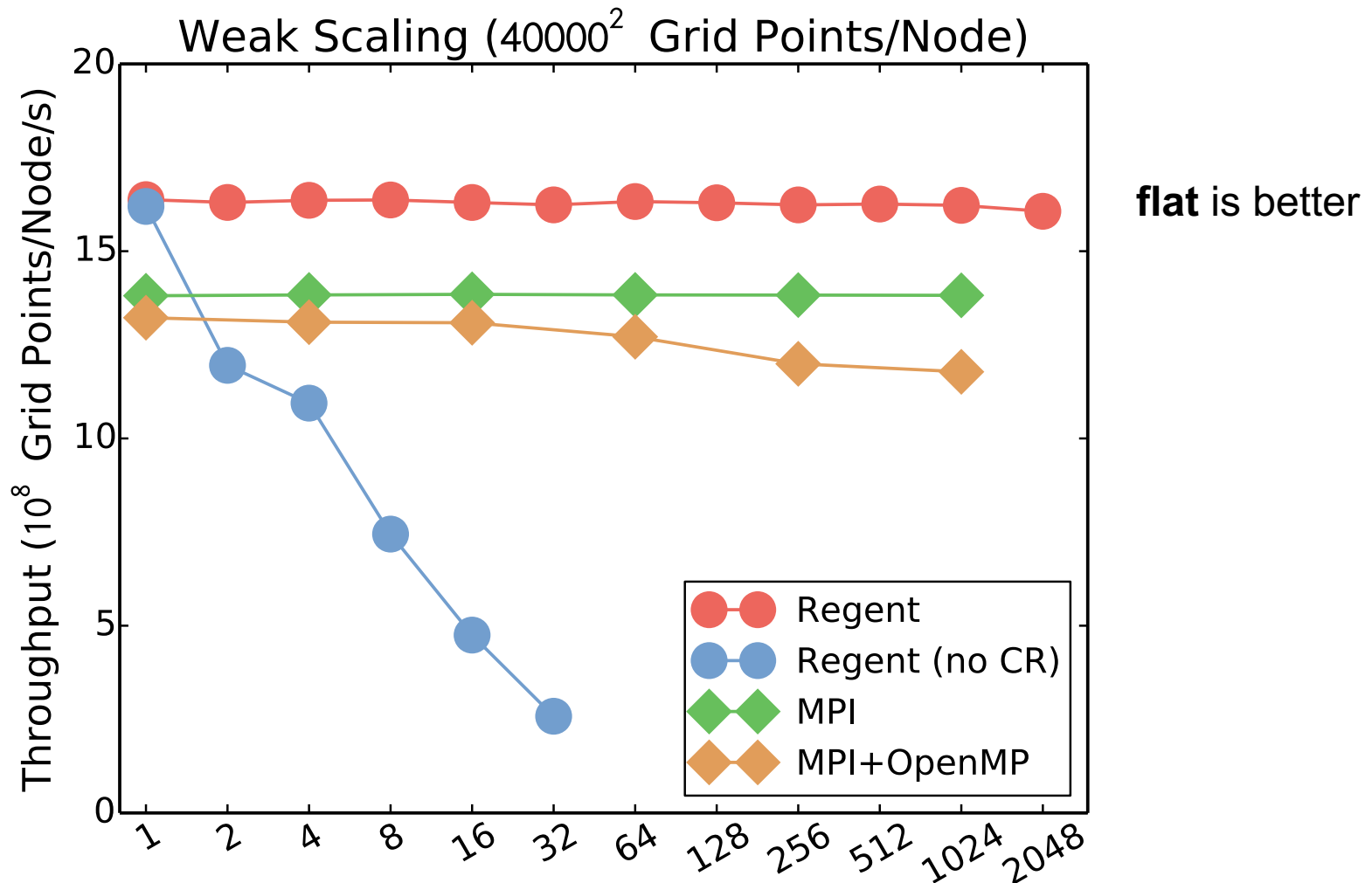
Goal: Pushing the Performance Envelope with Compilation



Static Control Replication

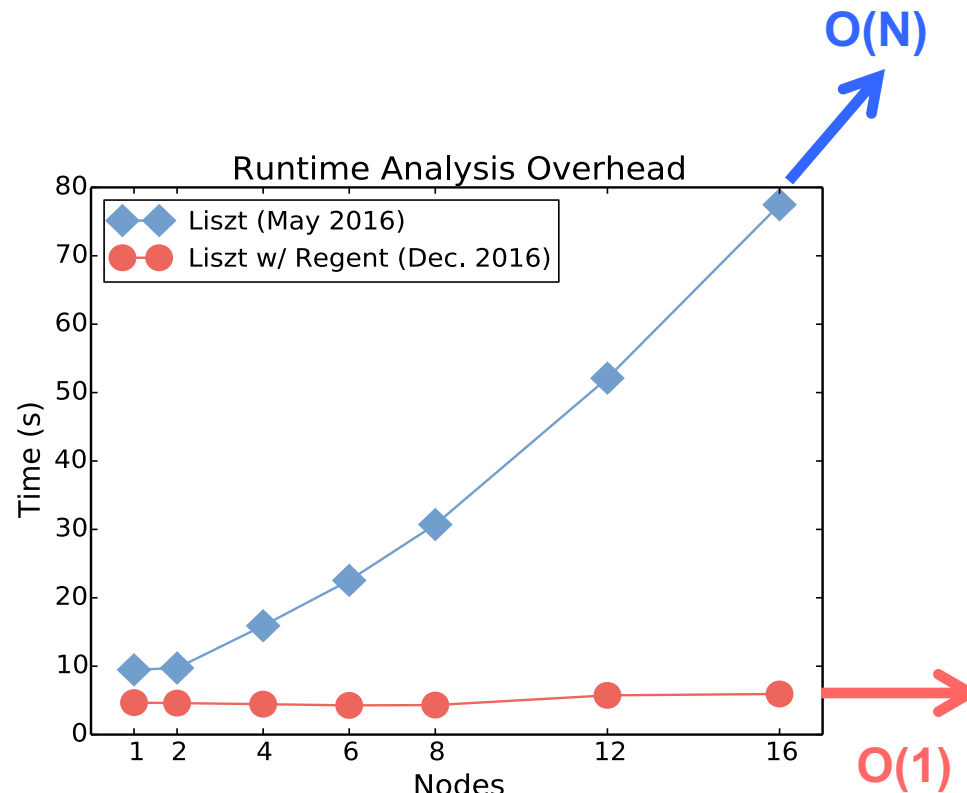


Weak Scaling with Control Replication



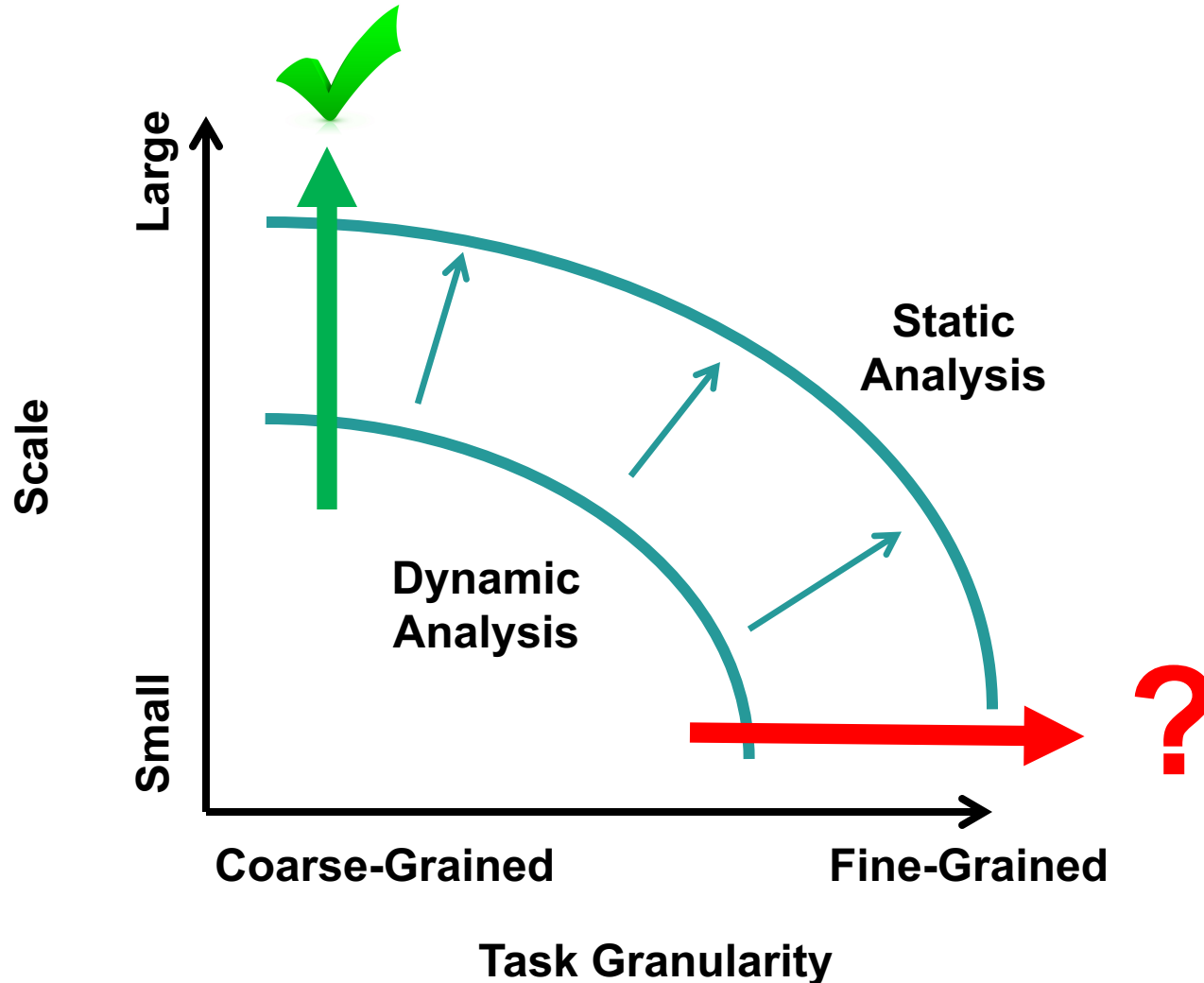


Impact of Control Replication: $O(N)$ Overhead $\Rightarrow O(1)$



lower is better

Goal: Pushing the Performance Envelope with Compilation

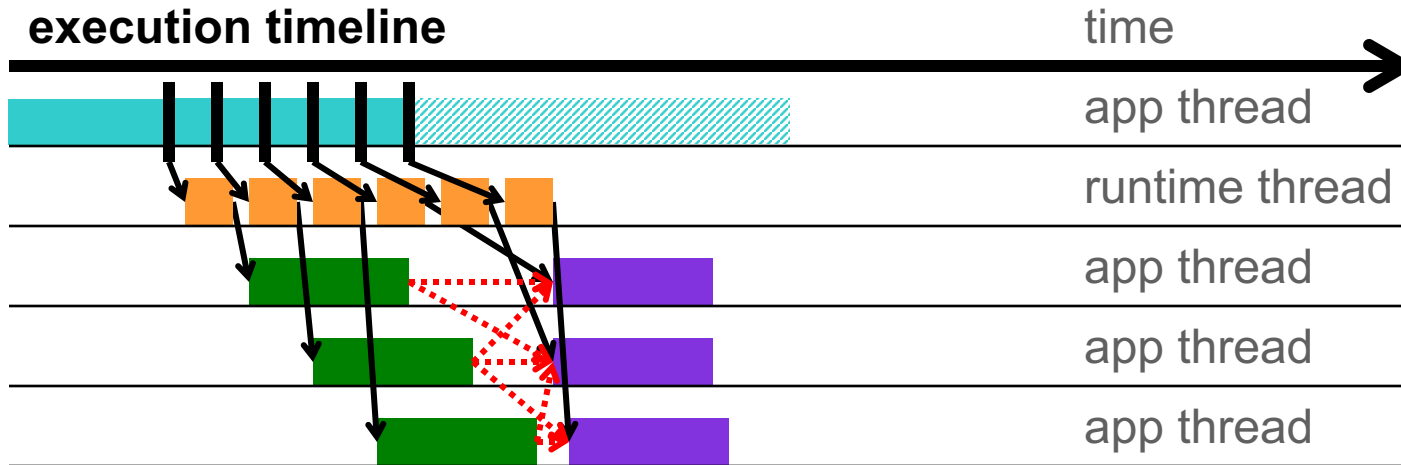




Why Static Analysis?

- Legion is a dynamic **pipelined** runtime
 - Logical dependence analysis
 - Mapping
 - Physical dependence analysis
 - Execution
- Cost is hidden as long as:
 - Throughput of runtime \geq velocity of tasks
- Use static analysis to avoid work at runtime
 - Ideal case:
 - ~~Logical dependence analysis~~
 - **Mapping**
 - ~~Physical dependence analysis~~
 - Execution

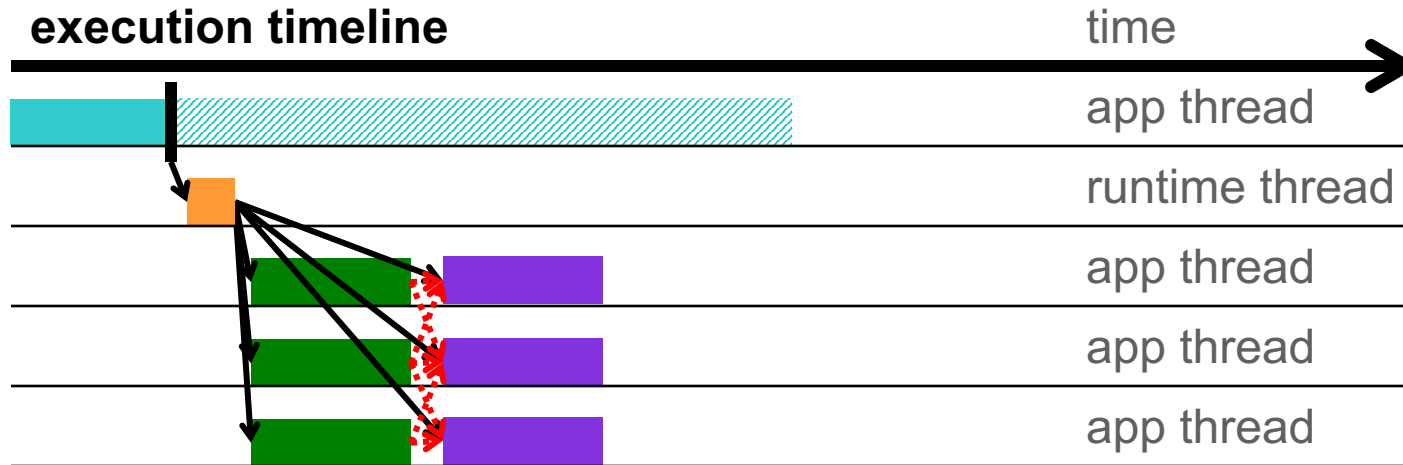
Today: Dynamic Dependence Analysis



```
for i = 0, 3 do
  calc_forces(..., points)
end
for i = 0, 3 do
  adv_pos_full(p_points[i])
end
```




Future: Static Dependence Analysis

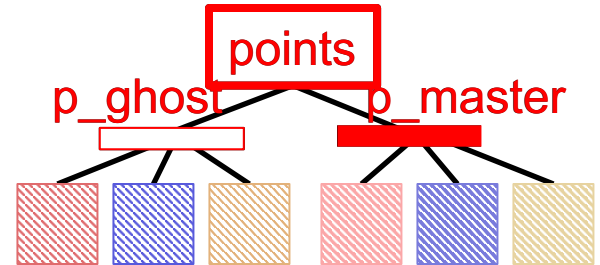


```
for i = 0, 3 do
  calc_forces(..., points)
end
for i = 0, 3 do
  adv_pos_full(p_points[i])
end
```

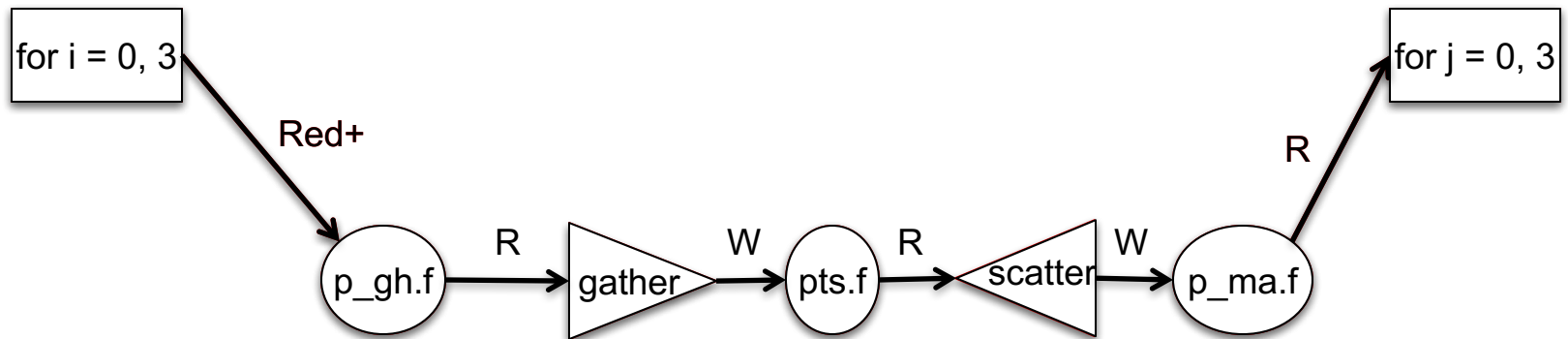


RDIR: Construction

```
while t < T do
  for i = 0, 3 do -- Red+(p_ghost)
    calc_forces(..., p_ghost[i]) -- Red+(p_ghost[i])
  end
  for j = 0, 3 do -- R(p_master)
    adv_pos_full(p_master[i]) -- R(p_master[i])
  end
  ...
end
```



while t < T





Plan for Static Analysis

- **Static Dependence Analysis (RDIR)**
- **Static Mapping (Bishop)**
- **Generate Static Realm Dataflow Graph**
- **Runs as Operation in Legion Pipeline**



Not Just Static: JIT

- **Some applications don't fit static analysis**
- **But some of these properties are JIT-static**
- **Start executing Regent compiler at runtime**
 - **Just another stage in the runtime**