# Parallel Dynamic Programming

Kenneth L. Judd, Hoover Institution
Yongyang Cai, Hoover Institution
Greg Thain, University of Wisconsin-Madison
Stephen Wright, University of Wisconsin-Madison

May 29, 2012

# Parallel DP Algorithm

- Parallelization in Maximization step in NDP: Compute

$$v_i = \max_a \; u_t(x_i, a) + \beta E\{\hat{V}(x^+; \mathbf{b}^+)|x_i, a\},$$

  for each $x_i \in X$, $1 \leq i \leq m$.

- Condor Master-Worker system: distributed parallelization, two entities: Master processor, a cluster of Worker processors.

# Parallel DP Algorithm for Master

**Initialization.** Set up $\hat{V}(x, \theta; \mathbf{b}^T)$ and initial guesses of actions $a$, for all $\theta \in \Theta = \{\theta_j = (\theta_{j1}, \ldots, \theta_{jk}) : 1 \leq j \leq N\}$. Choose the approximation nodes, $X_t = \{x_i^t = (x_{i1}^t, \ldots, x_{id}^t) : 1 \leq i \leq m_t\}$ for $t = 0, 1, \ldots, T-1$. Let $t = T-1$.

**Step 1.** Separate the maximization step into $N$ tasks, one task per $\theta_j \in \Theta = \{\theta_j = (\theta_{j1}, \ldots, \theta_{jk}) : 1 \leq j \leq N\}$. Each task contains the parameters $\mathbf{b}^{t+1}$, and initial guesses of actions $a$ for all $x_i \in X_t$ with a given $\theta_j$. Then send these tasks to the workers.

**Step 2.** Wait until all tasks are done by the workers. Then collect the parameters $\mathbf{b}_j^t$ and optimal actions $a_{ij}^*$ from the workers, for $1 \leq i \leq m_t$ and $1 \leq j \leq N$.

**Step 3.** Stop if $t = 0$; else go to step 1.

# Parallel DP Algorithm for Worker

Step 1. Receive the parameters $\mathbf{b}^+$ and initial guesses for actions for one specific $\theta_j$ from the master.

Step 2. For $\theta_j$, compute

$$v_{ij} = \max_a \; u(x_i, \theta_j, a_{ij}) + \beta E\{\hat{V}(x_i^+, \theta_j^+; \mathbf{b}^+) \mid x_i, \theta_j, a\},$$

for each $x_i \in X_t$, $1 \le i \le m_t$.

Step 3. Using an appropriate approximation method, compute the $\mathbf{b}_j$, such that $\hat{V}(x, \theta_j; \mathbf{b}_j)$ approximates $\{(x_{ij}, v_{ij})$: $1 \le i \le m_t\}$.

Step 4. Send $\mathbf{b}_j$ and optimal actions $a_{ij}^*$ for $1 \le i \le m_t$, to the master.

# Parallelization in Optimal Growth Problems

- Problem size: 4D continuous state $k$, 4D discrete state $\theta$ with $6^4 = 1296$ values
- Performance:

| | |
|---|---|
| Wall clock time for all 3 VFIs | 65 hours |
| Total time workers were up (alive) | 1487 hours |
| Total cpu time used by all workers | 1358 hours |
| Minimum task cpu time | 557 seconds |
| Maximum task cpu time | 4,196 seconds |
| Number of (different) workers | 25 |
| Overall Parallel Performance | 93.56% |

# Parallelization in Optimal Growth Problems

**Parallel efficiency for various number of worker processors**

| # Worker processors | Parallel efficiency | Average task CPU time (minute) | Total wall clock time (hour) |
|---|---|---|---|
| 25 | 93.56% | **21** | **65** |
| 54 | 93.46% | **25** | **33** |
| 100 | 86.73% | **25** | **19** |

# Parallelization in Dynamic Portfolio Problems

Problem size: 6 stocks plus 1 bond, transaction cost, number of task = 3125.

- Performance:

| | |
|---|---|
| Wall clock time for all 6 VFIs | 1.56 hours |
| Total time workers were up (alive) | 295 hours |
| Total cpu time used by all workers | 248 hours |
| Minimum task cpu time | 2 seconds |
| Maximum task cpu time | 395 seconds |
| Number of (different) workers | 200 |
| Overall Parallel Performance | 87.2% |