Universität Zürich[UZH]

Hybrid parallel programming and DP – hands-on examples

Simon Scheidegger
simon.scheidegger@gmail.com
August 3rd, 2017
Open Source Macroeconomics Laboratory – BFI/UChicago

Including adapted teaching material from books, lectures and presentations by
B. Barney, B. Cumming, W. Gropp, G. Hager, M. Martinasso, R. Rabenseifner, O. Schenk, G. Wellein

Outline

I. Discrete state dynamic programming

→ OMP & MPI

→ MPI groups

1. DSDP – the model

$$V_{new}(k,\Theta) = \max_{c} \left(u(c) + \beta \mathbb{E}\{V_{old}(k_{next},\Theta_{next})\}\right)$$

$$\text{s.t.} \quad k_{next} = f(k,\Theta_{next}) - c$$

$$\Theta_{next} = g(\Theta)$$

States of the model:

- k : today's capital stock → **There are many independent k's**
- Θ: today's productivity state → **The Θ's are independent**

Choices of the model:

- k_{next}

→ k, k_{next}, Θ and Θ_{next} are limited to a finite number of values

solver.cpp – the critical loops

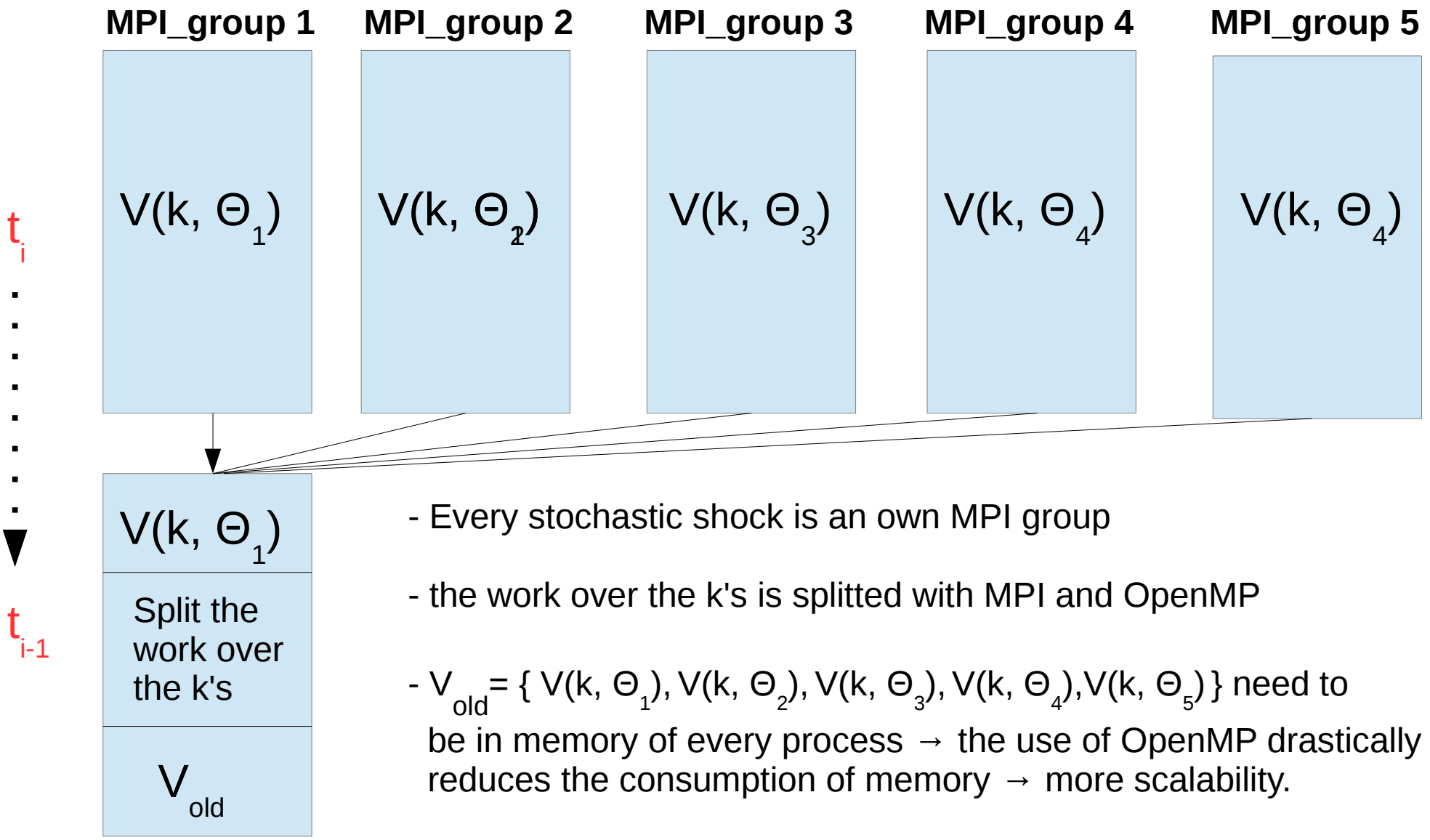
```
for (int itheta=0; itheta<ntheta; itheta++) {

    /*
     Given the theta state, we now determine the new values and optimal policies corresponding to each
     capital state.
     */

    for (int ik=0; ik<nk; ik++) {

        // Compute the consumption quantities implied by each policy choice
        c=f(kgrid(ik), thetagrid(itheta))-kgrid;

        // Compute the list of values implied implied by each policy choice
        temp=util(c) + beta*ValOld*p(thetagrid(itheta));

        /* Take the max of temp and store its location.
         The max is the new value corresponding to (ik, itheta).
         The location corresponds to the index of the optimal policy choice in kgrid.
         */
        ValNew(ik, itheta)=temp.maxCoeff(&maxIndex);

        Policy(ik, itheta)=kgrid(maxIndex);

    }
}
```

2). *split MPI communicator

1). distribute k's via OpenMP & MPI

loops to worry about

The parallelization scheme



- Every stochastic shock is an own MPI group

- the work over the k's is splitted with MPI and OpenMP

- $V_{old} = \{ V(k, \Theta_1), V(k, \Theta_2), V(k, \Theta_3), V(k, \Theta_4), V(k, \Theta_5) \}$ need to be in memory of every process → the use of OpenMP drastically reduces the consumption of memory → more scalability.

Let's do that together in class

→ OMP & MPI
→ MPI groups