Prof. Dr. Gerhard Zumbusch
Wissenschaftliches Rechnen II – Sommer 2023

4. Übungsblatt

Thema: Verteilter Speicher mit BSP

Aufgabe 4

Bulk Synchronous Parallel (BSP) ist ein Programmiermodell für verteilten Speicher mit ursprünglich nur einseitiger Kommunikation (get und put) und gleichzeitig ein theoretisches Performance-Modell. Ein Programm wird in Super-Steps zerlegt, die jeweils aus einem Rechenschritt und anschließend einem Kommunikationsschritt bestehen. Bei der Modellierung werden im Rechenteil wie üblich Operationen gezählt und im Kommunikationsteil die zu verschickende Datenmenge.

Ein BSP Programm in C++ besteht aus Initialisierung und dem Vereinbaren von Variablen (hier als C++ Iteratoren) und die anschließenden Super-Steps aus C++ Code mit Rechenoperationen, get und/oder put Befehlen und einem abschließenden sync. Ein Hello-World auf 4 Threads mit der BSPLib (siehe `http://bsplib.eu` und `https://github.com/Zefiros-Software/BSPLib`) könnte so aussehen:

```
#include <iostream>
#include <bsp/bsp.h>

int main(int, char**) {
  BSPLib::Execute( [] {
    std::cout << "Hello, this is process " << BSPLib::ProcId()
              << " of " << BSPLib::NProcs() << std::endl;
    BSPLib::Sync(); },
    4 );
}
```

Übersetzen mit `g++ -lpthread` und geeignetem include-Pfad `-I`. Variablen und Speicherbereiche müssen vor get/put in C++ erst einmalig registriert werden, etwa mit `BSPLib::PushIterator(..);`. Man könnte sich Programmiersprachanbindungen von BSP vorstellen, bei denen Registrierung und Initialiserung nicht nötig sind.

Zwischen den `BSPLib::Sync();` kann mit `BSPLib::PutIterator(..);` und/oder `BSPLib::GetIterator(..);` ein Datenaustausch angestossen werden, der im folgenden Super-Step, also nach dem nächsten `BSPLib::Sync();` dann auch ausgeführt ist.

Stellen Sie die verteilte Jacobi-Iteration aus den vorhergehenden Beispielen auf BSP um. Verwenden Sie get, um für jede Iteration die nötigen Daten an den Gitterpunkten vorliegen zu haben.