Prosabeschreibung der Datanaufteilung:

Die Matrix wird wie folgt aufgeteilt, da es sich um reihenbasierte
verfahren handelt bekommt jeder Thread (Reihenanzahl / Threadanzahl)
Reihen zugewiesen

Parallelisierungsschema Jacobi:

Bei Jacobi gilt es das von einer Matrix auf die nächste geschrieben wird.
Jedes Feld der neuen Matrix benötigt die werte von der alten Matrix
die auf der gleichen Feld Position und die die um einen Index wert drum herum
liegen. Sprich auch Werte wo die Zuständigkeit bei einem anderen Thread liegt.

dem entsprechend müssen alle Threads ihre erste Reihe der ersten Matrix
dem Thread der im Rank vor ihnen liegt schicken und und ihre letzte Reihe dem
Thread der im Rang nach ihnen liegt.

Ausnahmen bilden hier Rang 0 und Rang n (Mit n der letzte Rang bei N Threads).
Rang 0 hat keinen Vorgänger, somit muss er nur seine letzte Reihe an seinen
Nachfolger schicken. Rang n hat keinen Nachfolger und muss somit nur seine
erste Reihe an seinen Vorgänger schicken.

nach jeder iteration (wenn alle Felder in Matrix 2 gefüllt sind) muss einmal
gesynct werden sodas die reihen für die iteration dann von der Matrix 2 neu
aufgeteilt werden. und Matrix 1 dann wieder mit den neuen ergebnissen befüllt
werden kann.

So werden auch die Daten immer am anfang jeder Iteration ausgetauscht.

Parallelisierungsschema Gauß-Seidel:

Hier handelt es sich um ein Verfahren das auf einer matrix arbeitet. Es geht auch
reihenweise vor. Bei der initalisierung wird jedem Thread seine Reihe zugewiesen und
Jeder Prozess sendet seinem Vorgänger die erste Reihe. (bis auf ThreadRang 0, da
dieser keinen Vorgänger besitzt)

Danach fängt der Algorithmus in Reihe 1 an und geht iterativ durch alle durch.
Daraus folgt das jeder thread (bei Rang0 beginnend) immer die letzte Reihe nach seiner
Berechnung an seinen nachfolger schickt und und die erste an seinen Vorgänger. (Mit Rang 0
kann wieder beginnen sobald er von seinem Nachfolger die Reihe erhählt und muss auch
nur die letzte Reihe weitersenden. Rang n muss nur die erste zeile an seinen Vorgänger
schicken wenn er seine Berechnung beendet hat.

Der Clou hierbei ist. Die Vorgänger können immer schon anfangen sobald sie die Werte von ihrem
Nachfolger erreichen. obwohl die Iteration des Algorithmus noch nicht vollständig abgeschlossen ist.
Es ist keine Synchronisation nötig um den Algorithmus forzuführen.

Abbruch problematik:

Jacobi

Beide Abbruchbedingungen können nach der vollständigen Erstellung der Lösungsmatrix in jeder
Iteration überprüft werden. Da sowohl dann eine überprüfung stattfinden kann ob die Iterationen
vollständig abgeschlossen ist als auch das man eine Lösungsmatrix hat in der komplett geprüft werden
kann
ob alle Werte der Genauigkeit entsprechen. Die macht jeder Thread für seine Reihen und danach gibt es
einen
weiteren Sync, das entweder die Lösung den Abbrechbedingungen genügen oder eine weitere Iteration
durch
geführt werden muss.

Gauß Seidel

Die Abbruchbedingung der iterationsanzahl wird in Rang 0 geprüft und höhrt dann einfach auf wenn diese
erfüllt sind.

Bei der Abbruchbedingung der Genauigkeit is dies etwas schwieriger da wir ein Asynchrones verhalten
haben.
Da mehr iterationen das Ergenis vom Prinzip her nur noch genauer machen kann Thread Rang0 keine
Iteration
mehr anfangen wenn ihm alle anderen Threads zugesendet haben wenn sie genau genug sind und Thread 0
auch

genau genug ist um der Genauigkeits bedingung zu genügen.

Die letzte iteration läuft dann noch durch, damit alle Teilergebnisse aus der gleichen iteration kommen.
Ob für einen Thread die Genauigkeitsbedingung ausreichend erfüllt is. prüft jeder Thread für sich nachdem
ein neues Ergebnis errechnet hat.