Einführung

Eine Autofabrik steht still, weil die Ersatzteile nicht eingetroffen sind. Nach der Ankunft des Transports wird dieser abgeladen und die Produktion wird gestartet. Währenddessen steht der Transporter in der Halle und setzt sich erst wieder in Bewegung, als die Fließbänder mangels Rohstoffen zum Stillstand kommen. Der Einwand mag nun berechtigt sein, dass es doch effizienter wäre, wenn der Transporter schon wieder neue Rohstoffe holt, während die Produktion noch läuft um einen reibungslosen und vor allem stillstandlosen Ablauf zu gewährleisten. Computersysteme mit einem Kern arbeiten genau nach dieser Prämisse. Jeder Aufgabe wird streng sequentiell bearbeitet. Bei verschiedenen Betriebssystemen wie zum Beispiel Microsoft Windows XP oder dem Linux-Kernel wird eine Illusion von Parallelität mithilfe des Multi-Taskings erreicht, bei der Programme abwechselnd den Zugriff auf die Hauptrecheninheit erhalten. In der Summe dauert durch den ständigen Kontextwechsel die Berechnung bei mehreren Prozessen jedoch sogar noch länger, als wenn jeder Prozess für die gesamte Rechendauer exklusiven Zugriff auf das Rechenwerk hätte. Parallele Programmierung tritt an, diese Problematik entgegenzutreten und eine stark parallelisierte Welt computergestützt abzubilden.

Fazit

Wie gezeigt wurde, bietet parallele Programmierung die Möglichkeit weit effizienter Probleme der realen Welt zu lösen, als es dies mit sequentieller Programmierung auf Einprozessorsystem möglich wäre. Dies ist jedoch mit den genannten Einschränkungen verbunden, dass es auch Problemstellungen gibt, die sich kaum oder garnicht parallelisieren lassen. Ausserdem wird ist das Programmiermodell komplexer und aufgrund dessen fehleranfälliger und mit mehr Programmieraufwand verbunden.