



## **Diplomarbeit: (Selbst-)Adaptive Zustandspartitionierung replizierter verteilter Objekte**

Im Rahmen des EU-Projekts XtreamOS wird hier in Ulm eine Bibliothek zur Unterstützung *Virtueller Knoten (Virtual Nodes, VN)* entwickelt. Ein Virtueller Knoten repräsentiert dabei eine Gruppe von Prozessen, die jeweils die gleiche Anwendung repliziert ausführen. Durch diesen Mechanismus wird die Anwendung fehlertolerant, da der Ausfall eines oder mehrerer Knoten nicht zum Ausfall der Anwendung führt. Ein Nachteil von Fehlertoleranzmechanismen sind die teilweise hohen Laufzeitkosten. Diese sind ein Folge davon, dass viele Nachrichten totale geordnet werden müssen - jeder der replizierten Prozesse sieht diese Nachrichten in der gleichen Reihenfolge.

Die totale Ordnung ist besonders dann hinderlich, wenn zwei Nachrichten geordnet werden, bei denen dies überhaupt nicht nötig ist. Dies ist zum Beispiel der Fall, wenn beide Nachrichten eine Zustandsänderung auslösen, aber jeweils unterschiedliche Teile des Zustands geändert werden. Steht der Virtuelle Knoten unter hoher Last, können solche unnötige Sortierungen schnell zu einem Flaschenhals werden und der Durchsatz leiden. In einem solchen Fall, wäre es hilfreich Teile des Zustands und der Funktionalität des VNs in einen neuen VN auszulagern. Um möglichst wenig Kommunikation zwischen den beiden VNs zu benötigen sollten sowohl Zustand als auch Funktionalität des ausgelagerten VNs möglichst disjunkt zu dem Zustand und der Funktionalität sein, die im ursprünglichen VN verbleibt.

Da die für die Implementierung von VNs eingesetzte Sprache, Java, keine Kontrolle über die Zugriffe auf den Zustand bietet, ist es Teil der Arbeit einen solchen Kontrollmechanismus zu realisieren. Idealerweise sollte das bekannte Programmiermodell von Java erhalten bleiben, als auch Legacy-Anwendungen unterstützt werden. Weiterhin soll untersucht werden, ob und wie es möglich ist Teile des Codes wie zum Beispiel eine Funktion zur Laufzeit auszutauschen. Möglicherweise ist *aspect-orientiert programming (AOP)* ein geeigneter Ansatz.