Diss. ETH No. 12906

Scheduling for Heterogeneous Opportunistic Workstation Clusters

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY
ZURICH

for the degree of
Doctor of Technical Sciences

presented by
MATTHIAS NEERACHER
Dipl. Inf.-Ing., ETH Zürich
born 1 February 1967
citizen of Zürich

accepted on the recommendation of
Prof. Dr. W. Fichtner, supervisor
Prof. Dr. L. Thiele, co-examiner

1998

Abstract

In recent years, clusters of networked workstations have become an increasingly popular source of computing resources for large, computation and memory intensive calculations. Instead of setting aside dedicated workstations for such clusters, the spare computing resources of interactively used workstations can be harnessed into opportunistic clusters. Since equipment in workstation environments is often underutilized, opportunistic clusters offer the potential of obtaining considerable computation resources at essentially no additional hardware cost.

In this dissertation, I present the design and implementation of DMW, a system to schedule a collection of interdependent computing jobs on an opportunistic cluster consisting of workstations based on a variety of hardware and software platforms, connected by a local area network and sharing a common file namespace.

Designers of distributed systems often have to choose between fully distributed communication architectures and architectures with some of the communication and decision flow concentrated in a central component. Having implemented both a fully distributed and a partially centralized version of DMW, I contrast the two approaches, concluding that the partially centralized approach offers better performance and global fairness and reduces communication overhead compared to the fully distributed approach.

Any evaluation of a scheduler for an opportunistic workstation cluster cannot rest solely on the computation performance offered to users of the scheduler: For such a scheduler to be accepted in a workstation environment, it is of primary importance that resource demands it places on a workstation never interfere with the needs of interactive users using that workstation.

To analyze this problem, I present a series of experiments documenting the resource consumption patterns of jobs scheduled by the local scheduler on a workstation. I demonstrate that CPU use of background jobs is well controlled by local schedulers and rarely interferes with interactive users, but that memory use of background jobs is a serious problem which cannot be adequately addressed by local schedulers.

Since both the resource demands of background jobs and the resource availability on the target workstations is not predictable, a scheduler must be capable of migrating running jobs to other workstations if conditions change, i.e., the job becomes too large for its current workstation, an interactive user starts using the workstation, or another, more attractive workstation becomes available. While I did not implement a migration mechanism for DMW, I present design criteria for such a mechanism. In particular, I argue that an application independent mechanism is unsatisfactory in heterogeneous environments and requires impractical amounts of disk space and communication bandwidth for the large jobs typical for the intended use of DMW, and that an application specific migration mechanism can overcome these problems.

Zusammenfassung

In den letzten Jahren sind Verbunde vernetzter Workstation-Rechner zu einem immer beliebteren Mittel zur Bereitstellung von Rechenressourcen für grosse, rechen- und speicherintensive Berechnungen geworden. Dabei müssen die Rechner für einen solchen Verbund nicht eigens bereitgestellt werden, da es möglich ist, die brachliegenden Rechenkapazitäten interaktiv genutzter Rechner in opportunistischen Rechnerverbunden zu verwenden. Weil Computer in Workstation-Umgebungen oft nur wenig ausgelastet sind, versprechen opportunistische Verbunde, beträchtliche Rechenressourcen praktisch ohne zusätzliche Hardwarekosten zu erhalten.

In dieser Dissertation behandle ich den Entwurf und die Entwicklung von `DMW`, einem System, das den Ablauf einer Gruppe verknüpfter Rechenaufträge auf einem opportunistischen Verbund plant. Dieser besteht aus Rechnern, die auf verschiedenen Hardware- und Softwareplattformen basieren, mit einem Netzwerk verknüpft sind, und über ein gemeinsames Dateisystem verfügen.

Die Entwickler verteilter Systeme müssen oft eine Wahl treffen zwischen einer vollständig verteilten Systemarchitektur und einer Architektur, bei der Teile des Kommunikations- und Entscheidungsflusses in einer zentralen Komponente konzentriert sind. Anhand einer vollständig verteilten und einer teilweise zentralisierten Ausführung von `DMW` vergleiche ich die beiden Ansätze und komme zum Schluss, dass der teilweise zentralisierte Ansatz bessere Leistung und systemweite Fairness bringt und zudem im Vergleich zum vollständig verteilten Ansatz einen geringeren Kommunikationsaufwand benötigt.

Die Beurteilung eines Schedulers (Ablaufplaners) für einen opportunistischen Rechnerverbund kann sich nicht allein auf die Leistung stützen, die den Benutzern des Schedulers geboten wird: Für die Akzeptanz eines solchen Systems in einer Workstation-Umgebung ist es von entscheidender Bedeutung, dass die Ressourcenbedürfnisse, die es an einen Rechner stellt, niemals zu Lasten eines interaktiven Benutzers des Rechners gehen.

Um dieses Problem zu analysieren, zeige ich in einer Reihe von Experimenten auf, wie sich die Ressourcenbedürfnisse von Rechenaufträgen entwickeln, die auf einem einzelnen Rechner vom dortigen lokalen Scheduler verwaltet werden. Dabei wird ersichtlich, dass der Rechenbedarf von Hintergrundberechnungen vom lokalen Scheduler gut kontrollierbar ist und kaum je Probleme für interaktive Benutzer darstellt, dass aber der Speicherbedarf dieser Berechnungen ein schwerwiegendes Problem darstellt, das vom lokalen Scheduler nicht hinreichend gelöst werden kann.

Da sowohl die Ressourcenbedürfnisse von Hintergrundberechnungen als auch die zur Verfügung stehenden Ressourcen auf einem Rechner nicht vorhersehbar sind, muss ein Scheduler imstande sein, laufende Berechnungen auf andere Rechner zu verschieben, wenn sich die Bedingungen verändern, d.h. wenn die Berechnung zu gross für ihren gegenwärtig zugewiesenen Rechner wird, wenn ein interaktiver Benutzer den

Rechner zu belegen beginnt, oder wenn ein anderer, schnellerer Rechner frei wird. Ich habe keinen Migrationsmechanismus für DMW entwickelt, diskutiere jedoch Entwurfskriterien für einen solchen Mechanismus. Insbesondere argumentiere ich, dass ein programmunabhängiger Mechanismus für heterogene Umgebungen unbefriedigend ist und dass er für die grossen Rechenaufträge, die für den Einsatz von DMW typisch sind, unrealistische Mengen an Plattenspeicher und Kommunikationskapazität benötigt. Ein programmspezifischer Mechanismus ist dagegen in der Lage, diese Probleme zu überwinden.