Supercomputerzugang für Arbeitsplatzsysteme - Ein neuer Weg

SAVE-Tagung Travemünde

17. April 1991

Joachim Backes
Dr. Martin Bürkle
Marita Lange

Regionales Hochschulrechenzentrum
der Universität Kaiserslautern

Paul-Ehrlich-Straße
D-6750 Kaiserslautern

Inhaltsverzeichnis

1. Einführung

Die Akzeptanz eines Mainframe- oder Supercomputer-Betriebssystems durch den Benutzer hängt nicht nur von den Funktionen ab, die das System bietet. Diese sind mehr oder weniger interner Art.

Die Bereitstellung neuer Generationen leistungsfähiger Terminals einerseits sowie der Trend zum Einsatz lokaler Intelligenz (*Workstations, PCs*) andererseits gestatten nun auch die Diskussion von Fragen der *Ergonomie*: Gestaltung einer benutzerfreundlichen Oberfläche (mausgesteuerte Bedienung) und frühzeitige Kontrolle zur Vermeidung syn- taktischer und/oder semantischer Probleme.

Kommandointerpreter wie SDF (BS2000) oder ISPF bzw. PFD (MVS, VSP/I,...) beispielsweise zeigen zwar Ansätze in die richtige Richtung. Ihre Beschränkung auf relativ unintelligente Terminals wie IBM/3270 oder SNI/9750 lässt sie jedoch unattraktiv erscheinen.

Ein Supercomputer-Zugang, der mit einer Frontend-Backend-Konstruktion verbunden ist, d.h. mit inkompatiblen Kommandoschnittstellen, Betriebssystemmeldungen u.ä., verstärkt diese Problematik noch.

Ziel der vorliegenden Studie ist, einen Ansatz zur Lösung dieser Probleme aufzuzeigen, und zwar auf der Basis von UNIX-Workstations wie z.B. WS30 oder SUN. Als wesentlichche Aspekte seien dabei hervorgehoben:

- einfache Bedienung mit Maussteuerung und Window-Mechanismen, lokale Syntax- und Semantikkontrollen;
- portable Implementierung mit UNIX als Betriebssystem; Sockets, X-Windows, X-Inrinsics sowie OSF/Motif-Toolkit als darüber gelagerte Applikationsschichten;
- Durchgriff auf die kompletten Mainframe- bzw. Supercomputer-Funktionen;
- Kompatibilität zu USER2000, dem BS2000-Interface für Supercomputer der VP- und S-Serie;
- Objektorientiertheit mit weitgehender Unabhängigkeit von den Hostsystem-Eigenschaften: Dateidienste, Auftragsdienste, Informationsdienste,...;
- komplette variable Netzeinbindung.

2. Xvp - Ein Client-Server-Konzept für Supercomputer

Am Regionalen Hochschulrechenzentrum Kaiserslautem (RHRK) wird seit einigen Jah- ren ein Supercomputer vom Typ VP100 mit VSP/I als Betriebssystem eingesetzt. Benut- zerseitig wird der Zugang über die USER2000-Software gesteuert, einem BS2000-Dia- log- und Batchinterface auf der Basis von SDF (**S**ystem **D**ialog **F**acility), der BS2000-Standard-Kommandosprache.

Prinzipiell ist es hierbei durchaus möglich, dass sich ein irgendwo im Netz befindlicher User über TCP/IP im BS2000-Frontend mit *telnet* anmeldet oder über *ftp* mit dem BS2000 und damit dem VP Daten austauscht. Aber:

- die Verwendung von USER2000 setzt einige Kenntnis des BS2000 voraus, insbesondere auch der Kommandosprache SDF;
- SDF, der Editor und speziell USER2000 stellen eine völlig ungewohnte Umgebung für Unix- und Workstation-erfahrene Anwender dar.

Hieraus ergibt sich, dass es wesentlich sinnvoller ist, auf der *Workstation* eine einheitliche Oberfläche (im folgenden **Xvp** [**X**-Windows access to **v**ector *p*rocessor] genannt) zur Verfügung zu stellen, die nur geringen Bezug auf die speziellen BS2000-Eigenschaften nimmt.

Ein sehr angenehmer Nebeneffekt ergibt sich daraus fast zwangsläufig: Hat man einmal eine solche Oberfläche, so lässt sich diese leicht für den Zugriff auf andere Systeme portieren, insbesondere auch direkt auf Supercomputer. Denn: die funktionellen Unter- schiede zwischen den einzelnen Betriebssystemen sind relativ gering.

Man kann davon ausgehen, dass TCP/IP als Transportinterface auf *allen* beteiligten Rechnern verfügbar ist und der Quasistandard X-Windows zumindest auf der benutzereigenen Workstation.

Die Aufgabenverteilung zwischen Workstation(s) einerseits und dem Host (beliebiger Mainframe oder Supercomputer) andererseits ist dann wie folgt festgelegt:

- Benutzereigene Workstation oder X-Terminal vor Ort: Der Bildschirm wird vom X-Server bedient. Bei einer Workstation kann hier der UNIX(-spezifische Anteil von Xvp (im folgenden XvpU genannt) laufen, muss aber dann portiert werden. Oder
- eine *Server-Workstation* mit XvpU: der User muss sich von seinem (X-)Terminal mit *telnet* auf dieser Workstation einloggen und XvpU starten mit der Randbedingung, dass XvpU mit seinem eigenem X-Server in Verbindung tritt (standardmäßig der Toolkit-Parameter "-display <eigene Internet-Adresse>:0". XvpU wandelt Nachrichten, die über die grafische Toolkit-Schnittstelle vom X-Server empfangen werden, in einen Byte-Strom um, der an den
- hostseitigen Anteil von Xvp (*XvpH* = benutzerspezifischer Agent) weitergereicht wird. XvpH wird von einem hostspezifischen Server XvpS über ein TCP/IP-Verbindungsprotokoll kreiert. XvpH ähnelt stark einem Transaktionsmonitor. Seine Aufgabe besteht im wesentlichen darin, von XvpU Nachrichten in einem Host-unabhängigen Format entgegenzunehmen, diese in vom Host verständliche Aktionen

zu übersetzen, die Aktionen auszuführen und die Ergebnisse zu XvpU zurückzusenden.

- XvpU schließlich wandelt die Antwort mittels Toolkit-Funktionen in grafische Protokollelemente um und übergibt sie an den X-Server vor Ort.

Das Kommando, mit dem im UNIX-Server XvpU aufgerufen wird, hat im wesentlichen folgendes Aussehen:

```
xvp   -display <eigene Internetadresse>:0
      -userid <hostspezifische Accounting-Angaben>
      -mainframe <Host-Internetadresse>
```

Also:

1. login an der eigenen Workstation
2. telnet zum Unix-Server
3. **xvp -display ... -userid ... -mainframe ...**

Abbildung 1 im Abschnitt "**Xvp-Funktionen und -Menü-Aufbau**" verdeutlicht das Ganze noch einmal.

Wenn man davon ausgeht, dass die X-Windows-, X-Toolkit- und OSF/Motif-Software auch im Host verfügbar sind, könnte man natürlich auch daran denken, die XvpU-Funktionen in den Host zu verlagern und mit XvpH zusammenzuführen. Es ist aber zu bedenken, dass dann praktisch jede Tastatur-Eingabe und Mauseingabe den Austausch von Protokoll-Elementen zwischen dem X-Client XvpU und dem X-Server in der User-Workstation nach sich ziehen - mit weitreichenden Konsequenzen, falls der Hostrechner ein Supercomputer ist!

3. Werkzeuge zur XvpU-Generierung

Unter UNIX windowfähige Programme zu schreiben, ist ein schwieriges Unterfangen. Gerade weil die Möglichkeiten, mit den X-Bibliotheken X-Library, X-Toolkit und OSF/MotifToolkit Programmentwicklung zu betreiben, praktisch unbegrenzt sind, ist umgekehrt der Programmierer angehalten, sämtliche X-Aufrufe selbst durchzukonstruieren, wozu die X-Software selbst kaum Hilfen anbietet. Insbesondere die Einpassung der unterschiedlichen Fenstertypen, Schriften, Füll- oder Randmuster in das Gesamtbild als "nicht leicht" zu beschreiben, ist noch untertrieben.

Mittlerweile werden von verschieden Firmen Software-Produkte angeboten, die einem solche mehr oder wenige formale Arbeiten abnehmen, so z.B.

- DIATOOLS
- DIALOG Builder
- iXBUILD
- DIALOG Manager

Sie erzeugen im wesentlichen einen mehr oder weniger portablen C-Code, der in das eigentliche Verarbeitungsprogramm für die Vorlaufphase einzubinden ist. Dabei hat sich her-

ausgestellt, dass die Produkte *DIATOOLS* und *DIALOG Builder* gemäß unseren Anforderungen nicht eingesetzt werden können, da die erzeugten C-Codes nur auf bestimmten Maschinen wie z.B. WS30, WX200 oder MX-Rechnern ablauffähig sind, teilweise nicht einmal X-Windows-fähig waren (lt. SNI soll von *DIATOOLS* bald eine X-fähige Version, jedoch nur für WS30 unter Apollo/Domain, freigegeben werden).

Nach eingehender Überprüfung hat sich die Xvp-Entwicklergruppe entschlossen, die bislang manuelle Vorgehensweise durch den Einsatz von *iXBUILD* (Fa. IXOS) zu automatisieren.

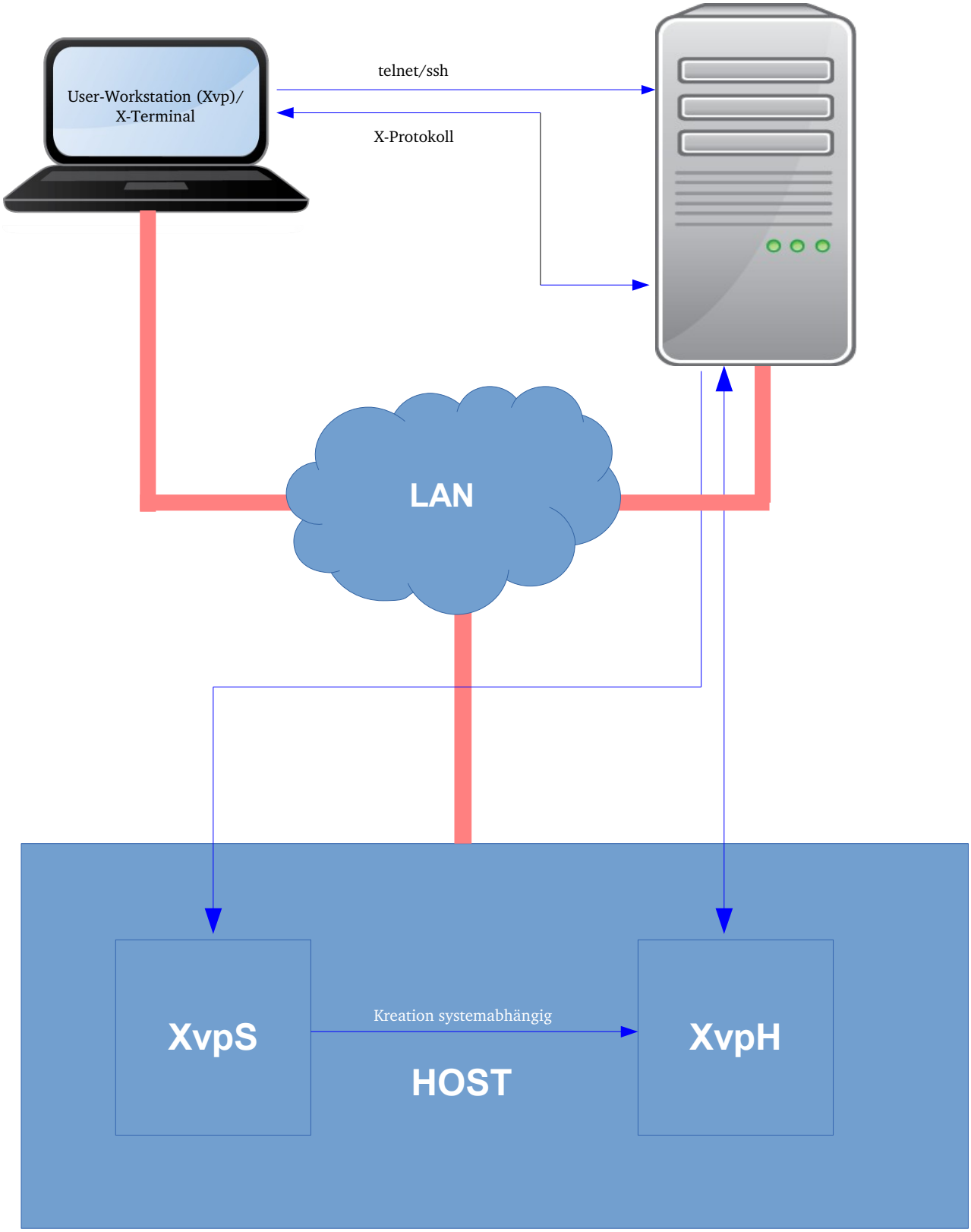

Abbildung 1: Netzeinbindung von Xvp

4. Xvp-Funktionen und -Menü-Aufbau

Das Xvp-Arbeitsfenster ist automatisch an die Bildschirmgröße der Benutzer-Workstation anpassbar. An Farben werden nur Grautöne bzw. Graumuster verwendet. Damit ist man unabhängig von den speziellen Eigenschaften des verwendeten Bildschirms.

Aufgeteilt ist das Xvp-Arbeitsfenster in sechs Bereiche:

- Eine Menüleiste mit folgenden Pulldown-Menüs:

  - **DESKTOP** → Programm-Kurzinformation
    → Xvp-Beendigung
  - **FILE** → Anzeigen von Datei-Informationen
    → Kreieren von Dateien
    → Löschen von Dateien
    → Kopieren von Dateien
    → Umbenennen von Dateien
    → Modifizieren von Dateikenndaten
    → Transfer von Dateien zwischen Workstation und Host
    → Weitere Hostsystem-abhängige Dateidienste, z.B.Kompression von Bibliotheken, Generieren von DD-Statements u.ä.

  Als Dateien sind, abhängig vom Hostsystem, eigentliche Dateien und Bibliotheken gemeint.

  - **PROGRAM** →Starten von Programmen im synchronen und im asynchronen Mo dus.

  Dabei verstehen wir unter synchronem Modus, dass gewartet wird, bis die Hostergebnisse vorliegen. Im asynchronen Modus ist dies nicht der Fall.

  - **JOB** → Starten von Jobs im synchronen (s.o.) und asynchronen Modus
    → Information über laufende Jobs
    → Abbrechen von Jobs
    → Starten von Job-Prozeduren im synchronen (s.o.) und im asynchro nen Modus. Unter Jobprozeduren verstehen wir den Start von para metrierten Jobs, d.h. der Jobinput enthält Formalparameter, die beim Jobstart durch Aktual-Parameter ersetzt werden.

  - **FORTRAN77**
    → Übersetzen von FORTRAN77-Programmen
    → Binden von FORTRAN77-Programmen
    → Optimierungsläufe von FORTRAN77-Programmen

  Dabei hängt letztendlich die gewählte Sprache vom Host-Betriebssystem ab; FORTRAN77 wurde gewählt, weil diese Sprache im technisch-wissenschaftlichen Bereich

und auf Supercomputern am weitesten verbreitet ist.

- **SYSTEM** ➞Reserviert für hostsystem-spezifische Anwendungen

- **EXPERT** ➞Starten einer Subshell
  ➞Sichern des Bildschirminhaltes in einer Datei (aus Dokumentati onsgründen)

- **MEMORY** ➞Anzeigen des Memory-Files
  ➞Wiederausführen einer im Memory-File abgespeicherten Aktivität entweder im Experten-Modus (s.u) oder Menü-gesteuert.

Jede ausgeführte Aktion, sei es im Experten- oder Window-Modus, wird in einem Memory-File, mit Zeitstempel versehen, abgelegt.

Dadurch hat der Benutzer einerseits eine Historie seinerAktivitäten zur Verfügung, andererseits kann er ausgewählte Aktionen erneut ausführen lassen.

- **HELP** ➞Help-Funktionen.

- Ein Fenster für den Expertenmodus. Expertenmodus heißt folgendes: Neben der menügesteuerten Eingabe können die in der Menüleiste verfügbaren Aktionen auch über UNIX-konforme-Kommandos ausgeführt werden, d.h.

  <*command*> [*operands*...] [-*options*...]

  Daneben wird zur Kontrolle nach einer Menü-Eingabe das entsprechende Expertenkommando in diesem Fenster abgelegt (und kann anschließend, wenn gewünscht, als Expertenkommando emeut ausgeführt werden). Sowohl die explizit eingegebenen Expertenkommandos als auch solche, die aus Menü-Eingaben resultieren, werden (s.o.) im Memory-File abgelegt.

  Die Namen der Kommandos sind so gewählt, dass für einen UNIX-Kenner die Zusammenhänge erkennbar sind: das Löschen einer Datei z.B. wird über das Kommando

  vrm file1 [file2...]

  initiiert (*v* wie vector, *rm* entsprechend dem bekannten UNIX-Kommando).

- Ein Fenster, in dem Nachrichten von XvpS und XvpH angezeigt werden (z.B. wichtige RZ-Mitteilungen, Gründe, aus denen XvpH nicht gestartet werden konnte, ...).

  Wegen des zu erwartenden Nachrichtenumfangs kann man in diesem Fenster über horizontale und vertikale Rollbalken scrollen.

- Ein Fenster, in dem nach erfolgter XvpH-Initialisierung ständig eine Liste der userspezifischen Files aufgezeigt wird. Diese Liste wird laufend auf dem neuesten Stand

gehalten: lässt der Benutzer über Xvp Dateien kreieren, löschen u.a., dann wird diese Liste automatisch aktualisiert.

Wegen des zu erwartenden Nachrichtenumfangs kann man in diesem Fenster über Rollbalken scrollen.

Da ein sinnvolles Arbeiten erst nach Eintreffen der Dateiliste möglich ist, sind die meisten Aktionen bis dahin gesperrt.

- Eine Reihe von Aktionsknöpfen: Klickt man in dem o.e. Filefenster Dateien (bzw. Bibliotheken) nach Belieben an, dann kann man durch zusätzliches Anklicken eines Aktionsknopfes Dateioperationen ausführen, die keine weiteren Angaben erfordern:

  - ➔ Löschen,
  - ➔ Ausgabe von Kenndaten,
  - ➔ Liste aktualisieren,
  - ➔ Hostsystemspezifische Aktionen wie z.B. das Komprimieren einer Bibliothek,
  - ➔ Liste reduzieren (insbesondere, wenn länger als Fenstergröße) und
  - ➔ reduzierte Liste wieder in der ursprünglichen Länge darstellen.

- Ein Automatismus verhindert unzulässige Operationen: selektiert man z.B. eine gewöhnliche Datei, oder gemischt Dateien und Bibliotheken, so werden die Aktionsknöpfe, die auf Bibliotheken operieren, deaktiviert. Dies resultiert aus der Xvp-Objektorientiertheit.

- Der Rest des Arbeitsfensters, d.h. im wesentlichen die linke Hälfte, dient der Darstellung der aktionsabhängigen Ergebnisfenster, welche der Übersicht halber untereinander versetzt dargestellt werden. Sie können unabhängig voneinander manipuliert werden. Die folgende Abbildung 2 zeigt für eine Beispielsitzung das Arbeitsfenster mit einem der oben beschriebenen Unterfenstern.

Abbildung 2: Anwendungsfenster von Xvp