To appear in: Euro-Par 2000 --- Parallel Processing, Arndt Bode and Thomas Ludwig (editors). To be presented at Euro-Par 2000, European Conference on Parallel Computing, Euro-Par 2000, Munich, Germany, August 29 - September 1, 2000

ETH Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Departement of Computer Science *Laboratory for Computer Systems*

Partition Cast — Modelling and Optimizing the Distribution of Large Data Sets in PC Clusters

Felix Rauch, Christian Kurmann and Thomas M. Stricker

Laboratory for Computer Systems
Swiss Federal Institute of Technology (ETH)
CH-8092 Zürich, Switzerland
{rauch,kurmann,tomstr}@inf.ethz.ch
http://www.inf.ethz.ch/
CS Technical Report #343

Abstract

Multicasting large amounts of data efficiently to all nodes of a PC cluster is an important operation. In the form of a *partition cast* it can be used to replicate entire software installations by cloning. Optimizing a partition cast for a given cluster of PCs reveals some interesting architectural tradeoffs, since the fastest solution does not only depend on the network speed and topology, but remains highly sensitive to other resources like the disk speed, the memory system performance and the processing power in the participating nodes. We present an analytical model that guides an implementation towards an optimal configuration for any given PC cluster. The model is validated by measurements on our cluster using Gigabit- and Fast Ethernet links. The resulting simple software tool, *Dolly*, can replicate an entire 2 GByte Windows NT image onto 24 machines in less than 5 minutes.

1 Introduction and Related Work

The work on partition cast was motivated by our work with the Patagonia multi-purpose PC cluster. This cluster can be used for different tasks by booting different system installations [12]. The usage modes comprise traditional scientific computing workloads (Linux), research experiments in distributed data processing (data-mining) or distributed collaborative work (Linux and Windows NT) and computer science education (Windows NT, Oberon). For best flexibility and maintenance, such a multi use cluster must support the installation of new operating system images within minutes.

The problem of copying entire partitions over a fast network leads to some interesting tradeoffs in the overall design of a PC cluster architecture. Our cluster nodes are built from advanced components such as fast microprocessors, disk drives and high speed network interfaces connected via a scalable switching fabric. Yet it is not

obvious which arrangement of the network or which configuration of the software results in the fastest system to distribute large blocks of data to all the machines of the cluster.

After in depth analytical modelling of network and cluster nodes, we create a simple, operating system independent tool that distributes raw disk partitions. The tool can be used to clone any operating system. Most operating systems can perform automatic installation and customization at startup and a cloned partition image can therefore be used immediately after a partition cast completes.

For experimental verification of our approach we use a meta cluster at ETH that unites several PC clusters, connecting their interconnects to a dedicated cluster backbone. This cluster testbed offers a variety of topologies and networking speeds. The networks include some Gigabit networking technology like SCI [7, 5], Myrinet [3] with an emphasis on Fast and Gigabit Ethernet [13]. The evaluation work was performed on the Patagonia subcluster of 24 Dell 410 Desktop PCs configured as workstations with keyboards and monitors. The Intel based PC nodes are built around a dual Pentium II processor configuration (running at 400 MHz) and 256 MB SDRAM memory connected to a 100 MHz front side bus. All machines are equipped with 9 GB Ultra2 Cheetah SCSI harddisk drives which can read and write a data stream with more than 20 MByte/s.

Partition cloning is similar to general backup and restore operations. The differences between logical and physical backup are examined in [8]. We wanted our tool to remain operating system and file system independent and therefore we work with raw disk partitions ignoring their filesystems and their content.

Another previous study of software distribution [9] presents a protocol and a tool to distribute data to a large number of machines while putting a minimal load on the network (i.e. executing in the background). The described tool uses unicast, multicast and broadcast protocols depending on the capabilities and the location of the receivers. The different protocols drastically reduce the network usage of the tool, but also prevent the multicast from reaching near maximal speeds.

Pushing the protocols for reliable multicast over unreliable physical network towards higher speeds leads to a great variation in the perceived bandwidth, even with moderate packet loss rates, as shown in [11]. Known solutions for reliable multicast (such as [6]) require flow-control and retransmission protocols to be implemented in the application. Most of the multicast protocol work is geared to distribute audio and video streams with low delay and jitter rather than to optimize bulk data transfers at a high burst rate.

The model for partition cast is based on similar ideas presented in the throughput-oriented copy-transfer model for MPP computers [14].

A few commercial products are available for operating system installation by cloning, such as *Norton Ghost* [1], *ImageCast* [2] or *DriveImagePro* [3]. All these tools are capable of replicating a whole disk or individual partitions and generating compressed image files, but none of them can adapt to different networks or the different performance characteristics of the computers in PC clusters. Commercial tools also depend on the operating- and the file system, since they use knowledge of the installed operating- and file systems to provide additional services such as resizing partitions, installing individual software packages and performing customizations.

An operating system independent open source approach is desired to support partition cast for maintenance in Beowulf installations [2]. Other applications of our tool could include *presentation-*, *database-* or *screen image cast* for new applications in distributed data mining, collaborative work or remote tutoring on clusters of PCs. An early survey about research in that area including video-cast for clusters of PCs was done in the Tiger project [4].

2 A Model for Partition-Cast in Clusters

In this Chapter we present a modelling scheme that allows to find the most efficient logical topology to distribute data streams.

2.1 Node Types

We divide the nodes of a system into two categories, active nodes which duplicate a data stream and passive nodes which can only route data streams. The two node types are shown in Figure 1.

[1] Norton Ghost©, Symantec, *http://www.symantec.com/*

[2] ImageCast©, Innovative Software Ltd., *http://www.innovativesoftware.com/*

[3] DriveImagePro©, PowerQuest, *http://www.powerquest.com/*

Figure 1: An active node (left) with an in-degree of 1 and an out-degree of 2 as well as a passive node (right) with an in- and out-degree of 3.

Active Node A node which is able to duplicate a data stream is called an active node. Active nodes that participate in the partition cast store the received data stream on the local disk.

An active node has at least an in-degree of 1 and is capable of passing the data stream further to one or more nodes (out-degree) by acting as a T-pipe.

Passive Node A passive node is a node in the physical network that can neither duplicate nor store a copy of the data stream. Passive nodes can pass one or more streams between active nodes in the network.

Partition cast requires *reliable* data streams with flow control. Gigabit Ethernet switches do only provide *unreliable* multicast facilities and must therefore be modelled as passive switches that do only route TCP/IP point-to-point connections. Incorporating intelligent network switches or genuine broadcast media (like Coax Ethernet or Hubs) could be achieved by making them active nodes and modelling them at the logical level. This is only an option for expensive Gigabit ATM switches that feature multicast capability on logical channels with separate flow control or for simple switches that are enhanced by a special end-to-end multicast protocol that makes multicast data transfers reliable.

2.2 Network Types

The different subsystems involved in a partition-cast must be specialized to transfer long data streams rather than short messages. Partitions are fairly large entities and our model is therefore purely bandwidth-oriented. We start our modelling process by investigating the topology of the physical network and taking a note of all the installed link and switch capacities.

Physical Network The physical network topology is a graph given by the cables, nodes and switches installed. The vertices are labeled by the maximal switching capacity of a node, the edges by the maximal link speeds.

The model itself captures a wide variety of networks including hierarchical topologies with multiple switches. Figure 2 shows the physical topology of the meta cluster installed at ETH and the topology of our simple sub-cluster testbed. The sub-cluster testbed is built with a single central Gigabit Ethernet switch with *full duplex* point-to-point links to all the nodes. The switch has also enough Fast Ethernet ports to accommodate all cluster nodes at the low speed. Clusters of PCs are normally built with simple and fast layer-2 switches like our Cabletron Smart Switch Routers. In our case the backplane capacity for a 24 port switch is at 4 GByte/s and never results in a bottleneck.

Our goal is to combine several subsystems of the participating machines in the most efficient way for an optimal partition-cast, so that the cloning of operating system images can be completed as quickly as possible. We therefore define different setups of logical networks.

Logical Network The logical network represents a connection scheme, that is embedded into a physical network. A spanning tree of TCP/IP connections routes the stream of a partition cast to all participating nodes. Unlike the physical network, the logical network must provide reliable transport and flow control over its channels.

Star A logical network with one central server, that establishes a separate logical channel to all n other nodes. This logical network suffers heavy congestion on the outgoing link of the server.

***n*-ary Spanning Tree** Eliminates the server-bottleneck by using an *n-ary spanning tree* topology spanning all nodes to be cloned. This approach requires active T-nodes which receive the data, store it to disk and pass it further to up to n next nodes in the tree.

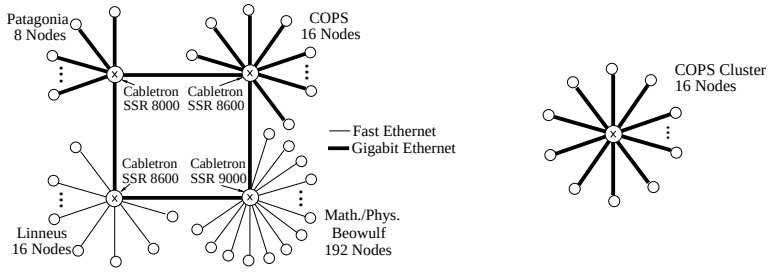

Figure 2: Physical network topologies of the ETH Meta-Cluster (left) and the simple sub-cluster with one central switch (right).

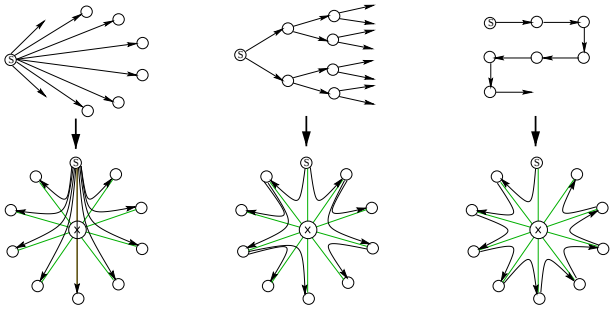

Figure 3: Logical network topologies (top) describing logical channels (star, n-ary spanning tree, multi-drop-chain) and their embedding in the physical networks.

Multi-Drop-Chain A degenerated, specialized tree (unary case) where each active node stores a copy of the stream to disk and passes the data to just one further node. The chain is spanning all nodes to be cloned.

Figure 3 shows the above described topologies as well as their embedding in the physical networks. We assume that the central switch is a passive node and that it cannot duplicate a partition cast stream.

2.3 Capacity Model

Our model for maximal throughput is based on capacity constraints expressed through a number of inequalities. These inequalities exist for active nodes, passive nodes and links, i.e. the edges in the physical net. As the bandwidth will be the limiting factor, all subsystems can be characterized by the maximal bandwidth they achieve in an isolated transfer. The extended model further introduces some more constraints e.g. for the CPU and the memory system bandwidth in a node (see Section 2.5).

Reliable transfer premise We are looking for the fastest possible bandwidth with which we can stream data to a given number of active nodes. Since there is flow control, we know that the bandwidth b of the stream is the same in the whole system.

Fair sharing of links We assume that the flow control protocol eventually leads to a stable system and that the links or the nodes, dealing with the stream, allocate the bandwidth evenly and at a precise fraction of the capacity.

Both assumptions hold in the basic model and will be slightly extended in the refined model that can capture raw and compressed streams at different rates simultaneously.

Edge Capacity defines a maximum streaming capacity for each physical link and logical channel (see Figure 4).

As the physical links normally operate in *full duplex mode*, the inbound- and outbound-channels can be treated separately. If the logical-to-physical mapping suggests more than one logical channel over a single physical link, its capacity is evenly shared between them. Therefore the capacity is split in equal parts by dividing the link capacity through the number of channels that are mapped to the same physical link.

Example: For a binary tree with in-degree 1 and out-degree 2 mapped to one physical Gigabit Ethernet link the bandwidth of a stream has to comply with the following edge inequality:

$$ {}_1E_2: \quad b < 125, \quad 2b < 125 \quad \rightarrow \quad b < \frac{125}{2} \tag{1} $$

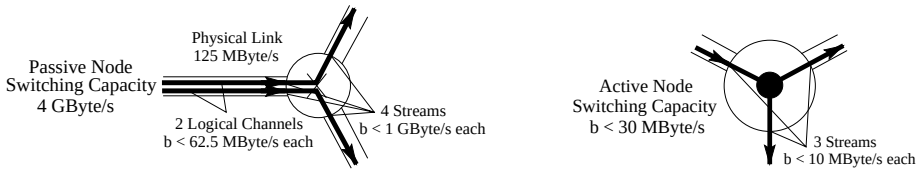

Figure 4: Edge capacities exist for the physical and logical network, node capacities for each in- and out-stream of a node.

Node Capacity is given by a switching capacity a node can provide, divided through the number of streams it handles.

The switching capacity of a node can be measured experimentally (by parameter fitting) or be derived directly from data of the node computer through a detailed model of critical resources. The experimental approach provides a specific limit value for each type of active node in the network, i.e. the maximal *switching capacity*. Fitting all our measurements resulted in a total switching capacity of 30 MB/s for our active nodes running on a 400 MHz Pentium II based cluster node. The switching capacity of our passive node, the 24 port Gigabit Ethernet switch is about 4 GByte/s - much higher than needed for a partition cast.

2.4 Model Algorithm

With the model described above we are now able to evaluate the different logical network alternatives described earlier in this Section of the paper. The algorithm for evaluation of the model includes the following steps:

```
algorithm basicmodel
1   chose the physical network topology
2   chose the logical network topology
3   determine the mapping and the edge congestions
4   for   all edges
          determine in-degree, out-degree of nodes attached to edge
          evaluate channel capacity (according to logical net)
5   for   all nodes
          determine in-degree, out-degree and disk transfer of the node
          evaluate node capacity
6   solve system of inequalities and find global minimum
7   return minimum as achievable throughput
```

Example: We compare a multi-drop-chain vs. the *n*-ary spanning tree structure for Gigabit Ethernet as well as for Fast Ethernet. The chain topology with all active nodes with in-degree *i* and out-degree *o* of exactly one (except for the source and the sink) can be considered as a special case of an unary tree (or hamiltonian path) spanning all the active nodes receiving the partition cast.

- *Topology:* We evaluate the logical *n*-ary tree topology of Figure 3 with 5 nodes (and a streaming server) mapped on our simple physical network with a central switch of Figure 2. The out-degree shall be variable from 1 to 5, multi-drop-chain to star.

- *Edge Capacity:* The in-degree is always 1. The out-degree over one physical link varies between 1 for the multi-drop-chain and 5 for the star which leads to the following inequality:

$$ {}_1E_o: \quad ob < 125 \quad \rightarrow \quad b < \frac{125}{o} \quad \text{for Gigabit Ethernet} \tag{2} $$

$$ {}_1E_o: \quad ob < 12.5 \quad \rightarrow \quad b < \frac{12.5}{o} \quad \text{for Fast Ethernet} \tag{3} $$

- *Node Capacity N:* For the active node we take the evaluated capacity of 30 MByte/s with the given in-degree and out-degree and a disk write:

$$ N_{1,o,1}: \quad (1+o+1)b < 30 \quad \rightarrow \quad b < \frac{30}{(1+o+1)} \tag{4} $$

We now label all connections of the logical network with the maximal capacities and run the algorithm to find a global minimum of achievable throughput. The evaluation of the global minimum indicates that for Gigabit Ethernet the switching capacity of the active node is the bottleneck for the multi-drop-chain and for the n-ary trees. But for the slower links of a Fast Ethernet the *n*-ary tree the network rapidly becomes a bottleneck as we move to higher branching factors. Section 4 gives a detailed comparison of modelled and measured values for all cases considered.

2.5 A more Detailed Model for an Active Node

The basic model considered two different resources: Link capacity and switching capacity. The link speeds and the switch capacity of the passive node were taken from the physical data sheets of the networking equipment, while the total switching capacity of an active node was obtained from measurements by a parameter fit. Link and switching capacity can only lead the optimization towards a graph theoretical discussion and will only be relevant to cases that have extremely low link bandwidth and high processing power or to systems that are trivially limited by disk speed. For clusters of PCs with high speed interconnects this is normally not the case and the situation is much more complex. Moving data through I/O buses and memory systems at full Gigabit/s speed remains a major challenge in cluster computing. The systems are nearly balanced between CPU performance, memory system and communication speed and some interesting tradeoffs can be observed. As indicated before, several options exist to trade off the processing power in the active node against a reduction of the load on the network. Among them are data compression or advanced protocol processing that turns some unreliable broadcast capabilies of Ethernet switches into a reliable multicast.

For a better model of an active node we consider the data streams *within* an active node and evaluate several resource constraints. For a typical client the node activity comprises receiving data from the network and writing partition images to the disk. We assume a “one copy” TCP/IP protocol stack as provided by standard Linux. In addition to the source and sink nodes the tree and multi-drop chain topologies require active nodes that store a data stream and forward one or more copies of the data streams back into the network. Figure 5 gives a schematic data flow in an active node capable of duplicating a data stream.

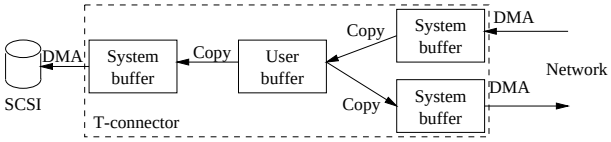

Figure 5: Schematic data flow of an active node running the *Dolly* client.

2.6 Modelling the Limiting Resources in an Active Node

The switching capacity of an active node is modelled by the two limits of the network and four additional resource limits within the active node.

Link capacity as taken from the physical specifications of the network technology (125 MB/s (Gigabit Ethernet) or 12.5 MB/s (Fast Ethernet) on current systems).

Switch capacity of passive nodes as taken from the physical specifications of the network hardware (2 or 4 GB/s depending on the Cabletron Smart Switch Router model, 8000 or 8600).

Disk system similar to a link capacity in the basic model (24 MB/s for a Seagate Cheetah 10'000 RPM disk).

I/O bus capacity the sum of data streams traversing the I/O bus must be less than its capacity (132 MB/s on current, 32 bit PCI bus based PC cluster nodes).

Memory system capacity the sum of the data streams to and from the memory system must be less than the memory system capacity (180 MB/s on current systems with the Intel 440 BX chipset).

CPU Utilization the processing power required for the data streams at the different stages. For each operation fraction coefficient $1/a_1, 1/a_2, 1/a_3, \ldots$ indicates the maximal speed of the stream with exclusive use of 100% CPU. The sum of the fractions of CPU use must be $< 1 (= 100\%)$ (Fractions considered: 80 MB/s SCSI transfer, 90 MB/s internal copy memory to memory, 60 MB/s send or receive over Gigabit Ethernet, 10 MB/s to decompress a data stream for a current 400 MHz single CPU cluster node).

Limitations on the four latter resources result in constraint inequalities for the maximal throughput achievable through an active node. The modelling algorithm determines and posts all constraining limits in the same manner as described in the example with a single switching capacity. The constraint over the edges of a logical network can be evaluated then into the maximum achievable throughput considering all limiting resources.

2.7 Dealing with Compressed Images

Partition images or multimedia presentations can be stored and distributed in compressed form. This reduces network load but puts an additional burden on the CPUs in the active nodes. Compressing and uncompressing is introduced into the model by an additional data copy to a *gunzip* process, which uncompresses data with an output data rate of about 10 MByte/s (see Figure 6).

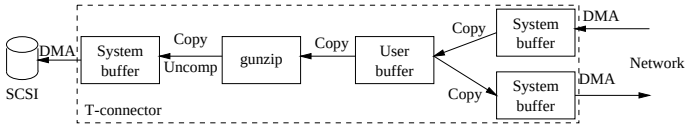

Figure 6: Schematic data flow of a *Dolly* client with data decompression.

The workload is defined in raw data bytes to be distributed and the throughput rates are calculated in terms of the uncompressed data stream. For constraints inequalities involving the compressed data stream the throughput must be adjusted by the compression factor c. Hardware supported multicast could be modeled in a similar manner. For multicast, the network would be enhanced by newly introduced active switches, but a reliable multicast flow control protocol module must be added at the enpoints and would consume a certain amount of CPU performance and memory system capacity (just like a compression module).

Example: Modelling the switching capacity of an active node for a binary-spanning tree with fast Ethernet and compression.

From the flow chart (similar to Figure 6 but with an additional second output stream from the user buffer to the network) we see two network sends, one network receive, one disk write, four crossings of the I/O bus, eleven streams from and to buffer memory, one compression module and five internal copies of the data stream.

This leads to the following constraints for the maximal achievable throughput b:

$$\begin{aligned}
\frac{b}{c} &< 12.5 \text{ MB/s} && \text{link for receive}\\
\frac{2b}{c} &< 12.5 \text{ MB/s} && \text{link for send}\\
b &< 24 \text{ MB/s} && \text{SCSI Disk}\\
\frac{3b}{c} + b &< 132 \text{ MB/s} && \text{i/o bus, PCI}\\
\frac{8b}{c} + 3b &< 180 \text{ MB/s} && \text{memory system}\\
(\frac{3}{45c} + \frac{1}{80} + \frac{4}{90c} + \frac{1}{90} + \frac{c}{9})b &< 1\ (100\%) && \text{CPU utilization}
\end{aligned}$$

For a compression factor of $c = 2$, an active node in this configuration can handle 5.25 MB/s. The limiting resource is the CPU utilization.

3 Differences in the Implementations

Our first approach to partition-cast was to use a simple file sharing service like NFS[4] with transfers over a UDP/IP network resulting in a *star* topology. The NFS server exports partition images to all the clients in the cluster. A command line script reads the images from a network mounted drive, possibly decompresses the data and writes it to the raw partitions of the local disk. Because of the asymmetric role of one server and many clients, this approach does not scale well, since the single high speed Gigabit Ethernet can be saturated even serving a small number of clients (see performance numbers in Section 4). Although this approach might look a bit naive to an experienced system architect, it is simple, highly robust and supported by every operating system. A single failing client or a congested network can be easily dealt with.

As a second setup, we considered putting together active-clients in a *n-ary spanning tree* topology. This method works with the standard TCP point-to-point connections and uses the excellent switching capability of the Gigabit Ethernet switch backplane. A partition cloning program (called *Dolly*) runs on each active node. A simple server program reads the data from disk on the image server and sends the stream over a TCP connection to the first few clients. The clients receive the stream, write the data to the local disk and send it on to the next clients in the tree. The machines are connected in an *n*-ary spanning tree, eliminating the bottleneck of the server link accessing the network.

Finally for the third optimal solution the same Dolly client program can be used with a local copy to disk and just one further client to serve. The topology turns into a highly degraded unary spanning tree. We call this logical network a *multi-drop chain*.

An obvious topological alternative would be a true physical spanning tree using the multicasting feature of the networking hardware. With this option the server would only source one stream and the clients would only sink a single stream. The protocols and schemes required for reliable and robust multicast are neither trivial to implement nor included in common commodity operating systems and often depend on the multicast capabilities of the network hardware. In a previous study [11] one of the authors implemented several well known approaches ([6, 10]). Unfortunately the performance reached in those implementation was not high enough to make an application to the partition cloning tool worthwhile.

4 Evaluation of Partition-Cast

In this section we provide measurements of partition casts in different logical topologies (as described in Section 2) with compressed and uncompressed partition images. The partition to be distributed to the target machines is a 2 GByte Windows NT partition. The compressed image file is about 1 GByte in size, resulting in a compression factor of 2.

A first version of our partition-cast tool uses only existing OS services and therefore applies a simplistic star topology approach. It consists of a NFS server which exported the partition images to all the clients. The clients accessed the images over the network using NFS, possibly uncompressing the images and finally writing the data to the target partition. The results of this experiment are shown on the left side of Figure 7 (the execution time for each client machine is logged to show the variability due to congestion). The Figure shows two essential results: (1) The more clients need to receive the data, the more time the distribution takes (resulting in a lower total bandwidth of the system). The bandwidth is limited by the edge capacity of the server. (2) Compression helps to increase the bandwidth for a star topology. As the edge capacity is the limiting factor, the nodes have enough

[4] Network File System

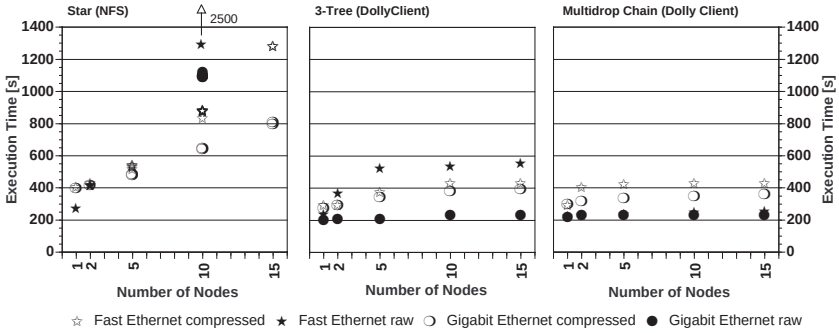

Figure 7: Total execution times for distributing a 2 GByte Windows NT Operating System partition simultaneously to 1, 2, 5, 10, 15 machines by partition cloning with NFS based star topology, the Dolly based 3-tree and multi-drop-chain topologies on the Patagonia cluster. The star topology run with 10 clients using raw transfers over Fast Ethernet resulted in execution times around 2500 seconds as the NFS servers disk started thrashing.

CPU, memory and IO capacity left to uncompress the incoming stream at full speed, thereby increasing the total bandwidth of the channel.

A second approach is to use an *n*-ary spanning tree structure. This topology was implemented in the small program *Dolly* which acts as an active node. The program reads the partition image on the server and sends it to the first *n* clients. The clients write the incoming data to the target partition on disk and send the data to the next clients. The out-degree is the same for all nodes (if there are enough successor-nodes) and can be specified at runtime. The results for a 3-ary tree are shown in Figure 7 in the middle. For Fast Ethernet the execution time increases rapidly for a small number of clients until the number of clients (and therefore the number of successor-nodes of the server) reaches the out-degree. As soon as the number of clients is larger than the out-degree, the execution times stay roughly constant. For this network speed, the edge capacity is again the bottleneck, resulting in increasing execution times for higher out-degree. In the case of Gigabit Ethernet, the link speed (the edge capacity) is high enough to satisfy an out-degree of up to 5 without the edge capacity becoming the bottleneck. The bottleneck in this case is still the nodes memory capacity.

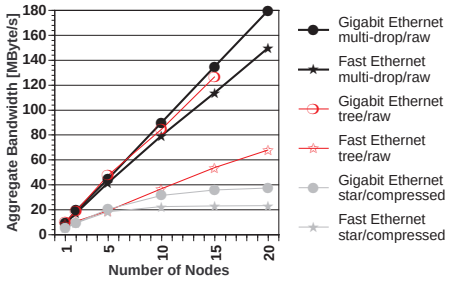

Figure 8: Total (aggregate) transfer bandwidth achieved in distributing a 2 GByte Windows NT Operating System partition simultaneously to a number of hosts by partition cloning in the Patagonia cluster.

For the third experiment we use Dolly to cast the partition using a multi-drop chain. The results are shown in Figure 7 on the right. They indicate that the execution time for this partition cast is nearly independent on the number of clients. This independence follows from the fact that with the multi-drop chain configuration, the edge capacity is no longer a bottleneck as every edge carries at most one stream. The new bottleneck is the nodes' memory system. The memory bottleneck also explains why compression results in a lower bandwidth for the channel (decompressing data requires more memory copy operations in the pipes to the gunzip process).

Figure 8 shows the total, aggregate bandwidth of data transfers to all disk drives in the system with the three experiments. The figure indicates that the aggregate bandwidth of the NFS-approach increases only modestly with the number of clients while the multi-drop chain scales perfectly. The 3-ary-tree approach also scales perfectly, but

increases at a lower rate. The numbers for the NFS approach clearly max out with the transfer bandwidth of the servers network interface reaching the edge capacity: Our NFS-server can deliver a maximum of about 20 MByte/s over Gigabit Ethernet and about 10 MByte/s over Fast Ethernet (note that we are using compressed data for the NFS approach in the above figure thereby doubling the bandwidth). The predicted bandwidths are compared with measured values in our cluster in Table 1.

Network			Fast Ethernet Bandwidth		Gigabit Ethernet Bandwidth	
Topology	Out-Degree	Compression	Extended Model	Measured	Extended Model	Measured
Multi-Drop-Chain	1	no	11.1	8.8	11.1	9.0
Multi-Drop-Chain	1	yes	6.1	4.9	6.1	6.1
2-Tree	2	no	6.3	5.4	8.1	8.2
3-Tree	3	no	4.2	3.8	6.4	8.0
Star	5	no	2.5	2.3	5.3	6.3
Star	5	yes	5.0	3.6	5.0	4.1

Table 1: Predicted and measured bandwidths for a partition cast over a logical chain and different tree topologies for uncompressed and compressed images. All values are given in MByte/s.

5 Conclusion

In this paper we investigated the problem of a partition-cast in clusters of PCs. We showed that optimizing a partition-cast or any distribution of a large block of raw data leads to some interesting tradeoffs between network parameters and node parameters.

In a simple analytical model we captured the network parameters (link speed and topology) as well as the basic processor resources (memory system, CPU, I/O bus bandwidth) at the intermediate nodes that are forwarding our multicast streams. The calculation of the model for our sample PC cluster pointed towards an optimal solution using uncompressed streams of raw data, forwarded along a linear multi-drop chain embedded into the Gigabit Ethernet. The optimal configuration was limited by the CPU performance in the nodes and its performance was correctly predicted at about one third of the maximal disk speed. The alternative of a star topology with one server and 24 clients suffered from heavy link congestion at the server link, while the different *n*-ary spanning tree solutions were slower due to the resource limitations in the intermediate nodes, that could not replicate the data into multiple streams efficiently enough. Compression resulted in a lower network utilization but was slower due to the higher CPU utilization. The existing protocols for reliable multicast on top of unreliable best-effort hardware broadcast in the Ethernet switch were not fast enough to keep up with our multi-drop solution using simple, reliable TCP/IP connections.

The resulting partition casting tool is capable of transferring a 2 GByte Windows NT operating system installation to 24 workstations in less than 5 minutes while transferring data at a sustained rate of about 9 MByte/s per node. Fast partition cast permits the distribution of entire installations in a short time, adding flexibility to a cluster of PCs to do different tasks at different times. A setup for efficient multicast also results in easier maintenance and enhances the robustness against slowly degrading software installations in a PC cluster.

References

[1] Henri Bal. The Distributed ASCI Supercomputer (DAS). *http://www.cs.vu.nl/~bal/das.html*.

[2] D. J. Becker, T. Sterling, D. Savarese, J. E. Dorband, U. A. Ranawake, and C. V. Packer. Beowulf: A Parallel Workstation for Scientific Computation. In *Proceedings of 1995 ICPP Workshop on Challenges for Parallel Processing*, Oconomowc, Wisconsin, U.S.A., August 1995. CRC Press.

[3] Nanette J. Boden, Robert E. Felderman, Alan E. Kulawik, Charles L. Seitz, Jakov N. Seizovic, and Wen-King Su. Myrinet — A Gigabit per Second Local Area Network. *IEEE-Micro*, 15(1):29–36, February 1995.

[4] William J. Bolosky, Joseph S. Barrera III, Richard P. Draves, Robert P. Fitzgerald, Garth A. Gibson, Michael B. Jones, Steven P. Levi, Nathan P. Myhrvold, and Richard F. Rashid. The Tiger Video Fileserver. In *Sixth International Workshop on Network and Operating System Support for Digital Audio and Video*, Zushi, Japan, April 1996. IEEE Computer Society.

[5] Dolphin Interconnect Solutions. *PCI SCI Cluster Adapter Specification*, 1996.

[6] S. Floyd, V. Jacobson, S. McCanne, L. Zhang, and C-G. Liu. A Reliable Multicast Framework For Lightweight Sessions and Application Level Framing. In *Proceedings of ACM SIGCOMM '95*, pages 342–356, August 1995.

[7] H. Hellwagner and A. Reinefeld, editors. *SCI Based Cluster Computing*. Springer, Berlin, Spring 1999.

[8] Norman C. Hutchinson, Stephen Manley, Mike Federwisch, Guy Harris, Dave Hitz, Steven Kleiman, and Sean O Malley. Logical vs. Physical File System Backup. In *Proceedings of the 3rd Symposium on Operating Systems Design and Implementation, New Orleans, Louisiana*, pages 239–249. The USENIX Association, February 1999.

[9] Steve Kotsopoulos and Jeremy Cooperstock. Why Use a Fishing Line When you Have a Net? An Adaptive Multicast Data Distribution Protocol. In *Proccedings of the USENIX 1996 Annual Technical Conference*, San Diego, California, January 1996. The USENIX Association.

[10] Sanjoy Paul, Krishan K. Sabnani, and David M. Kristol. Multicast Transport protocols for High Speed Networks. In *Proceedings of International Conference on Network Protocols*, pages 4–14. IEEE Computer Society Press, 1994.

[11] F. Rauch. Zuverlässiges Multicastprotokoll. Master's thesis, ETH Zürich, 1997. English title: Reliable Multicast Protocol. See also http://www.cs.inf.ethz.ch/. Contains a survey about reliable IP multicast.

[12] Felix Rauch, Christian Kurmann, Thomas Stricker, and Blanca Maria Müller. Patagonia — A Dual Use Cluster of PCs for Computation and Education. In *2. Workshop Cluster Computing, Karlsruhe*, March 1999.

[13] Rich Seifert. *Gigabit Ethernet: Technology and Applications for High-Speed LANs*. Addison-Wesley, May 1998. ISBN: 0201185539.

[14] T. Stricker and T. Gross. Optimizing Memory System Performance for Communication in Parallel Computers. In *Proc. 22nd Intl. Symposium on Computer Architecture*, pages 308–319, Santa Marguerita di Ligure, June 1995. ACM.