

Zeus Technology

Building High Performance, High-Availability Clusters

Date: 16th June 2000
Version: 1.0

Zeus Technology
Newton House
Cambridge Business Park
Cowley Road
Cambridge
CB4 0WZ
England
Phone: +44 (0) 1223 525000
Fax: +44 (0) 1223 525100
Email: info@zeus.com
Web: <http://www.zeus.com>

Table of Contents

| | |
|--|----|
| 1 Load Balancing Theory and Principles..... | 3 |
| 1.1 Types of load balancing..... | 3 |
| Round Robin DNS balancing..... | 3 |
| Simple randomisation..... | 3 |
| Response measurement..... | 3 |
| Content type..... | 4 |
| Geographical-based load balancing..... | 4 |
| 1.2 Session tracking..... | 4 |
| 2 Practical Approaches..... | 4 |
| 2.1 Round Robin DNS uses and caveats..... | 4 |
| 2.2 Hardware Load Balancers..... | 5 |
| 2.3 Zeus Load Balancer..... | 6 |
| 2.4 Advantages and disadvantages of each approach..... | 6 |
| 3 Load Balancing Implementation..... | 7 |
| 3.1 Terminology..... | 7 |
| Clustering..... | 7 |
| Load balancing..... | 7 |
| Fault-tolerance..... | 7 |
| Front end machines..... | 7 |
| Back end machines..... | 7 |
| 3.2 Worked example using Zeus Load Balancer..... | 7 |
| Essential components of a load-balanced system..... | 7 |
| The Administration Server..... | 7 |
| How many backend web servers to use?..... | 8 |
| Efficient network setup..... | 8 |
| IP addresses..... | 9 |
| 3.3 Configuring the Load-Balanced Cluster..... | 9 |
| Configuring a cluster of web servers..... | 9 |
| Installing the Load Balancers..... | 10 |
| Configuring load balancer failover..... | 10 |
| Example Load-balanced Cluster..... | 10 |
| How the back-end servers handle requests..... | 10 |
| 3.4 Monitoring the status of your cluster..... | 11 |
| 3.5 E-commerce..... | 12 |
| URL mappings..... | 12 |
| Cookies..... | 12 |
| Rewriting Configuration..... | 12 |

1 Load Balancing Theory and Principles

Load Balancing serves two distinct purposes in web-serving environments:

1. Prevent web servers from becoming overloaded, because of too much work. Spreading the workload of one website (or multiple websites) over several machines can lead to huge performance improvements, especially where the web servers are delivering dynamic content or running other CPU-intensive tasks. Buying lots of smaller machines is far cheaper than paying for one high-performance, specialist server.
2. Provide redundancy and failover protection. If one machine crashes, or suffers a hardware fault, then one of the other computers in the load-balanced cluster can step in and take over the workload. Visitors to the website will not experience any downtime. Since a load-balanced web server solution is distributed, adding or modifying hardware can be performed quickly and easily, without necessitating downtime.

1.1 Types of load balancing

The high-level concept of load balancing can be explained easily. When a webpage request is received, you must decide which web server should handle the task and deliver the content. However, there are many methods to use in selecting an appropriate web server. Here are the most common methods:

Round Robin DNS balancing

Very simple load balancing can be implemented using properties of the DNS. When a web browser visits a website, (e.g. www.zeus.com), it first contacts a DNS server in order to convert the 'www.zeus.com' part into the numerical IP address, which specifies the actual computer to contact and ask for a webpage from.

More than one IP address can be given for each site. When the browser queries the DNS to find the address of a site, the DNS server returns one of the available addresses, effectively chosen at random. So, multiple users of your website are likely to get given different addresses, so their page requests query different machines.

Using the DNS to load balance is unique in that the load balancing decisions are taken by each individual browser. All other solutions involve making a load balancing decision at the server side, using either a hardware or software solution. The load balancer itself is termed the 'frontend' of the website, and is visible to users. The web servers themselves are called 'backends' and are effectively hidden behind the load balancer.

Simple randomisation

A load balancer will just pick one of the available web servers at random and sends the webpage request onto it. Easy to implement, but randomisation is not as effective as one would hope. Even if all your web servers have comparable hardware, it will take just a few time-consuming requests directed to the same web server to cause that server to be swamped by too much work.

Randomisation should include the ability to detect for failed web servers and to avoid them as appropriate. Any modern load balancer will do this. Clearly, this ability alone places this method of load balancing ahead of round-robin DNS approaches.

Response measurement

The Load Balancer sends each webpage request to the web server it deems least busy.

There are lots of ways in which the load balancer can determine which web servers are working the most. The idea is to prevent any one web server from being overloaded, and so minimising response times for users.

Content type

Some load balancing solutions allow balancing based upon content. For example, several web servers could be setup to exclusively deliver images, whilst other machines concentrate on dynamic content. Other servers handle streaming video, and so on. Different servers, each optimised to provide a specific type of file or document, can provide better overall performance.

Geographical-based load balancing

Most load balancing techniques concentrate on balancing web page requests between web servers located on a LAN. However, the concept expands successfully to WAN-based load balancing as well. Requests for a web page can be redirected, on a per-page level, to a web server closer to the viewer.

1.2 Session tracking

One problem encountered when naively applying load-balancing solutions to a website is that of client state. Imagine a website implements a simple 'shopping-basket' facility, whereby customers can add products to their basket whilst continuing to browse the website. Now, the customers' choices must be stored on the server. This is simple when you have one machine running the web server. But if you load balance this website over lots of computers, keeping track of the shopping baskets becomes difficult.

Imagine a customer clicks on a product 'A' on the webpage. This request gets sent to the load balancer, which passes the request onto backend web server 1. Now, the customer clicks on product 'B'. This time, the request gets handled by web server 2. Web server 2 doesn't know anything about the customer's prior choices, so it thinks this customer's shopping basket only contains product 'B'. This will lead to a lot of confusion and unhappy customers!

The complex solution to this problem is to make your backend database aware of all the machines in the server farm. You could then run a single database server that gathers its information from all the running web servers. The web servers, in turn, use this database when generating pages. Unfortunately, this solution has lost some of the advantages that load balancing provides. We are back to a single point of failure (the database). The database application itself may now be the bottleneck, so performance advantages of multiple servers has been lost. Also, the database application itself must be carefully written so that it can cope with the possibility of its clients (the web servers) crashing or becoming unavailable. Likewise, adding machines to the system will be difficult.

The easy solution is to use 'session tracking'. Most load balancing systems will allow web servers to control the load balancing of individual users. Simply put, we ensure that all the webpage accesses of a single user will get sent to the same backend web server. The site is still load balanced because the page requests of all total users is still being spread over the server farm.

2 Practical Approaches

2.1 Round Robin DNS uses and caveats

By far the simplest load balancing mechanism, round-robin DNS is in wide use today. Unfortunately, it is not an ideal solution. Firstly, round-robin DNS leaves the load balancing decisions in the hands of client machines rather than at the server side, giving web-hosting providers little or no control.

There is no guarantee that your website traffic will be evenly spread over all the web servers in your farm, so many of your web servers could be wasted. Even worse, if the machine that a browser picks is not functioning, the user will just see a broken site. Losing web servers will lose you visitors.

Furthermore, DNS records are distributed and cached globally. This means that updating the DNS records, say to add another web server, will take at least a day before becoming effective. Adding new machines to cope with flash crowds is impossible.

Nevertheless, round-robin DNS is in very common use today, and shouldn't be dismissed out of hand. It can also prove useful when used in combination with other load balancing tools. This will be looked at later.

2.2 Hardware Load Balancers

There are many options for hardware load balancing nowadays. Companies such as Alteon, Cisco or Foundry Networks provide switches with load balancing capabilities built-in. With some configuration, you can then plug in machines running web servers into the switch, and traffic to them will be balanced between the running servers.

Just about all hardware balancers are described as 'layer 3' or 'layer 4' switches. Layer 3 switches look at the destination IP addresses of incoming requests and balance the requests on that information alone. Layer 4 switches look inside the TCP layer, and can selectively balance on port numbers (e.g. Balance only port 80 for HTTP).

Because all the load balancing is performed in special-purpose hardware, performance of the switches is extremely good. There are no practical limits on how many backend servers can be used.

Hardware load balancers are more intelligent than round-robin style approaches. No DNS tricks are required; the switch accepts all connections to a given address and then spreads these requests over the available machines. The switch will also monitor the backend machines for aliveness; if a machine dies, then the switch will stop allocating webpage requests to that computer until it is fixed.

The load balancing decisions taken by hardware switches are limited. They do not know the relative performance of each backend machine, and so for example they cannot give more work to an idle machine, for how can it tell which machines are least loaded?

Work allocation tends to be on a round-robin or random approach. Sometimes, selection is based on the machine with the least number of current connections, which gives some gain over random selection.

The hardware switch also cannot divert traffic based on URL, content-type or cookies, because this information can only be discovered by reading the actual HTTP traffic itself, not the basic underlying protocols. Slowly, level 5 switches are emerging, which offer some HTTP deciphering. However, they still lack the facilities that only a software-based approach can handle - this will be returned to later.

2.3 Zeus Load Balancer

The Zeus Load Balancer takes a software-based approach to load balancing. The product itself is a program running on a machine between the web servers and the Internet connection. Web page requests are directed to the machine(s) running the Load Balancer. Once the Balancer receives enough of the page request, it chooses the most appropriate web server to handle the request, and forwards on the data. The web server response passes back to the balancer and is then sent out onto the Internet.

The advantage of monitoring the web server responses as well as requests is that the Load Balancer can derive much more information this way than just letting the web servers send their content directly to clients. Firstly, response times are measured, to capture live performance information about each individual server. This allows the balancer to tell when an individual machine is heavily or lightly loaded and adjust future requests accordingly.

Further, because the Load Balancer effectively 'owns' all the web server connections, it can fully make use of 'keepalive' HTTP connections. These are a feature of HTTP 1.0 and 1.1 whereby an already-established connection may be used for future page fetches, saving large connection-setup costs and improving TCP traffic flow efficiency. This can give a noticeable performance boost to users of the website. Such techniques are impossible using traditional hardware load balancers.

The Load Balancer fully supports session management (as discussed earlier) as well as hard-coded balancing rules (e.g. Directing specific web pages to a single backend, for example where only that web server has the database or support programs to generate that type of page).

The Balancer runs on two machines in a 'master-master' relationship; both machines are in full use. If one machine were to fail, then the other will step in and take over its work. This gives an added layer of redundancy over simpler 'one box' approaches to load balancers. Such solutions end up moving the single point of failure from the web server onto the load balancer, giving a false sense of security.

Management of the load balancer is performed locally or remotely via the web-based Zeus Admin Server. This also allows live performance monitoring, to see if any web servers in the cluster have problems. Alerts about failures can be emailed or sent via SMS to network administrators.

2.4 Advantages and disadvantages of each approach

What are the disadvantages of such a software-based approach, compared to hardware load balancing? Firstly, hardware solutions offer a very quick-fit approach. There is little or no installation or software configuration required to get a simple balanced website up and running.

Moreover, the hardware is specialised for the purpose of load balancing and routing. Nowadays, if you buy a switch capable of load balancing, then its maximum throughput will almost certainly only be limited by the bandwidth of the media in use; 100Mb, 1Gbit or whatever. Software load balancers, since they work on generic hardware, will have performance depending upon the hardware that they run on. However, today's CPUs are easily fast enough to cope with all but the most extreme of conditions. Furthermore, since computers are getting faster and faster, the load balancer can be transferred to a quicker machine as and when required.

Software Load Balancers also commonly have the limitation that they can only balance specific network protocols. The Zeus Load Balancer works on the HTTP level of communication. While almost all hardware balancers don't understand HTTP because

they work at a lower level, it means that they can handle other protocols fairly easily. Services such as mail and FTP could all benefit from load balancing.

3 Load Balancing Implementation

3.1 Terminology

Many of the terms in these documents are commonly used to mean different things in computing literature, so an explanation of terms used is given first:

Clustering

Multiple machines with identical configurations (a cluster) to which requests can be made. The cluster appears to be one coherent unit; configuration changes made with the admin server are updated on all machines in the cluster simultaneously.

Load balancing

In the simplest form, this is just optimising incoming requests to the least busy machine in a cluster, although the Zeus Load Balancer performs other calculations in order to achieve the optimal response time for all users of the cluster.

Fault-tolerance

The transparent takeover by a machine of another failed machine, with continuity of service maintained. If a back-end server in a cluster fails, requests will no longer be sent to it until it returns to service. The twin front-end balancer configuration means service will continue if a front-end machine's hardware fails.

Front end machines

The Zeus Load Balancers. These are positioned at the 'front' of the cluster, i.e. they are the machines first contacted by web browsers requesting pages from the cluster.

Back end machines

The web servers running in the cluster. They are 'hidden' behind the Balancers, so that to each client web browser, the site appears to be one single webserver whereas it is in fact several machines.

3.2 Worked example using Zeus Load Balancer

Essential components of a load-balanced system

The basic architecture of the server farm consists of front-end machines, back-end machines, and a network connection to the outside world. The front-end machines receive requests from the Internet and distribute the requests to the back-end machines using the Zeus Load Balancer software.

The Administration Server

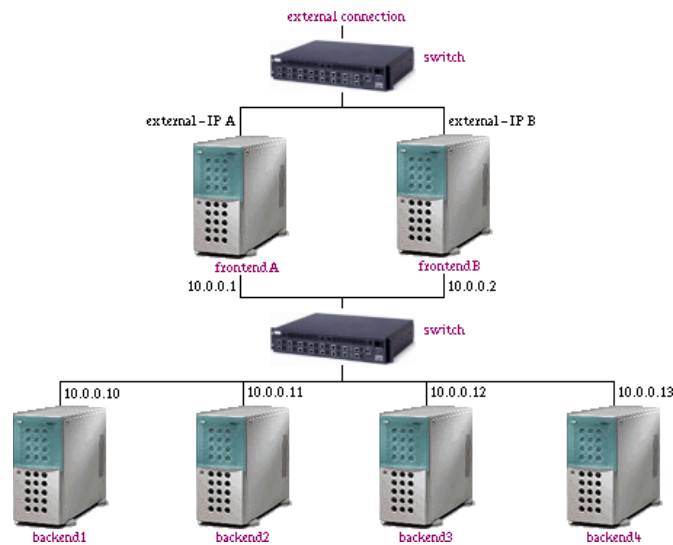
The Adminserver holds the master copy of configuration information for the websites and handles deploying this info across the machines in the server farm. The Adminserver is only required to be running when using the web-based admin interface to alter the configuration of virtual servers, it is not needed for the functioning of the system under normal operation.

How many backend webservers to use?

Any number of back-end machines can be used. If mostly static content will be served, and the Balancer is required more for fault-tolerance than performance, a minimal configuration of two back-end machines would be sufficient. If mostly complex dynamically generated content will be served, more back-end machines may be required to handle the load. The beauty of the Zeus Balancer system is that back-end machines can be dynamically added or removed as needs demand.

The front-end machines distribute requests to the back-end machines running in an active 'master-master' relationship, both funneling requests to the back-end machines. This gives both improved performance and fault-tolerance compared to using a single front-end machine.

Efficient network setup



The frontend machines each have two network cards. One is used solely to send and receive data to/from the outside world. The other is used to communicate with the backend webservers. Ensure that the routing on both frontend machines is configured correctly!

There are several points related to these setups that require discussion:

Only the front-end machines need be assigned an externally contactable IP address. The rest of the machines can use internal IP addresses. This will provide additional security - connections from outside should not be able to route to the backend machines. However, ensure that the adminserver is located so that it can still contact all machines in the cluster.

The network layouts discussed above do not consider the various means to supply content to the backend webservers. Such tasks are not the concerns of the Load Balancer itself! However, broadly speaking, there are two main methods for distributing website content - store a local copy on each of the backend machines and install some mechanism for synchronising the content on each server, or attach each backend to a network fileserver. In this case, be sure to consider where the fileserver will go in the network. The frontends do not need any connection with the filesystems.

Like with the frontend machines, if the backends are to communicate with a NFS server, then they can be fitted with two network cards - one solely for communicating with the frontends and the other for reaching the fileserver. This allows for maximum sustained throughput between the outside world and the fileserver.

IP addresses

Both frontend machines must have externally visible IP addresses. It is essential that the DNS for the websites is set up appropriately - the IP addresses for both machines should be listed in the DNS for the site, e.g. `www.cluster.zeus.com` has IP addresses 192.168.0.100 and 192.168.0.101. This enables round-robin DNS to function correctly - web browsers will pick one of the two addresses at random, and try to contact that individual machine.

If only one frontend is to be in active use, and the other is simply a backup, then obviously only one machine should appear in the DNS for the site.

Another fundamental issue is that, with two frontend machines, each machine must in fact have at least two IP addresses, even if they only have one network card! This is because during a failure, when one frontend dies, the other frontend must still have a valid IP address with which to try and contact the failed frontend, while at the same time masquerading as the failed frontend so that it receives all the HTTP requests.

This is better explained in terms of 'public' and 'private' IP addresses - do not confuse this with 'internal' and 'external' IP address ranges, there is a difference!

3.3 Configuring the Load-Balanced Cluster

Configuring a cluster of web servers

We will now run through the process of installation and configuration of a new Zeus Load-Balanced Cluster. Firstly, you will need to install the Zeus web server on all the backends; this can be done by running the `zinstall` script and selecting the 'Install Web Server only' option. Once installed, edit the `$ZEUSHOME/web/global.cfg` file, and add 'balancer!enabled yes' to this file, which tells the web server(s) to go into load balancer mode.

Once you have your web servers installed, copy the 'commkey' file from the Zeus admin server to the back-ends. This is located in `$ZEUSHOME/webadmin/conf/commkey` and should be copied to `$ZEUSHOME/web/etc/commkey`; this is a secret key which precedes all communication from the adminserver to the webserver cluster, allowing only the admin server to configure the web server cluster remotely.

Then call up the Zeus Admin server in your web browser, select 'Clustering' and proceed to add the backends by filling the in Configuration form. You should see a graphical tree grow for each backend you add on the main Cluster page.

Installing the Load Balancers

The Zeus Load Balancer can then be installed on the front-end machines, again by extracting the gzipped tar package and running the `zinstall` script, and entering the Load Balancer license key. You will again need to copy the 'commkey' file from the admin server to `$ZEUSHOME/balancer/etc/commkey` (permissions 400) so that the admin server can talk to the load balancers. You should also copy the `$ZEUSHOME/webadmin/conf/hosts` file from the admin server to `$ZEUSHOME/balancer/workers`. This file specifies the backends that the load balancer machines will use.

Return to the clustering page, and add the front-end load balancer to the cluster, by filling in the configuration form. You should see the front ends appear on the cluster tree diagram, and the servers should not be red. If they are, this means there is a communication problem; you should click the 'Summary' option and the admin server will diagnose the problem.

Configuring load balancer failover

The front-end load balancers should be configured to take over from each other, should one of them fail for any reason. This is determined by them monitoring the health of each other via 'ping' packets. If a response is not returned (by default) within 3 seconds, then failover will occur, and the remaining load balancer will take both front-end IP addresses.

To configure failover, the '`$ZEUSHOME/balancer/bin/configure-flipper`' script should be run from the command line. This will prompt you for various system details about your network and interfaces types: the name of each machine, the IP addresses, netmasks and broadcast addresses and the interface name for the network devices being used. This script creates a 'flipper.cfg' file; if this is placed in the `$ZEUSHOME/balancer` directory, a `zeus.flipper` process will be started when the 'start-zeus' script is next run which will perform the failover.

Example Load-balanced Cluster

Website `www.cluster.zeus.com` has two 'public' IP addresses, both in the DNS records - 192.168.0.100 and 192.168.0.101.

It also has two frontend machines to handle web page requests. These machines are `mercury` and `venus`. The IP addresses for `mercury` and `venus` are 192.168.0.2 and 192.168.0.3 respectively. These are the 'private' IP addresses.

In normal use, `mercury` and `venus` claim one of 192.168.0.100 / 101 each. If one frontend dies, then the other one claims both public IP addresses. In this way, all web page requests are received, even if one frontend stops working.



As discussed above, it is sensible to give the backend machines internal IP addresses. Remember that each network card interface requires a unique IP address, not just each machine.

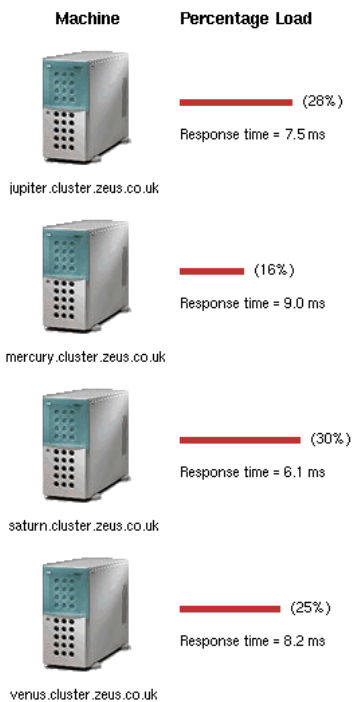
How the back-end servers handle requests

When you start a web site, the admin server sends the configuration to each of the web servers you have configured. Each web server then writes out a local copy of that configuration file in `$ZEUSHOME/web/runningsites`. This means the web server machine can reboot/restart without needing to communicate with the admin server.

3.4 Monitoring the status of your cluster

An icon is displayed for each machine configured in your cluster. These are grouped into two groups; front-end load-balancers and back-end servers. The front-end load-balancers only appear when the Zeus Load Balancer application is installed. A machine operating abnormally perhaps indicating a hardware failure will have a red icon. The icons for each box can be clicked on to provide more information. If you are using any non-Zeus backend webservers in a clustered install, they will also be shown on the page. However, because they lack the functionality of the Zeus Web Server, the admin server cannot check whether each non-Zeus server is working or not. You must check this for yourself.

| | |
|--|---|
|  | <p style="text-align: center;">Server operating normally</p> <p>This server can communicate with the admin server and is serving requests. Click on the server icon for more information.</p> |
|  | <p style="text-align: center;">Failed or incorrectly configured server</p> <p>The server has failed or the admin server cannot talk to it. This may happen if it has been incorrectly configured. Click on the server icon for more information.</p> |



The back-end web server machines also provide statistics, which can be viewed by clicking on a back-end server icon. They provide information on which sites being served are busiest, by default over a 5 minute interval.

Clicking on front-end machine icon will give a report on the percentage of traffic going to each back end, and the response time in milliseconds from this backend. Reloading this page will update the information.

One can also configure the Zeus Load Balancer to alert the system administrator if a machine goes down. A number of scripts are provided with the Load Balancer package which can be customised to hook into other alerting systems. The "alert-by-mail" script will send an email to the system administrator if one of the front-end machines or back-end machines is not responding.

3.5 E-commerce

In normal operation, the Zeus Load Balancer will function completely transparently to the web servers that it controls. However, there are occasions when programs or scripts that run on the web servers should have control over which backend machine is used for each request.

URL mappings

URL mapping is a powerful feature of the Zeus Load Balancer that enables particular web pages to be delivered by a specific machine in the cluster. This gives even greater control over the workload of each individual web server. Also, it enables clustering of websites where most of the content can be provided by all servers, but some specific areas must be handled by one machine.

Mapping works as follows; you specify a list of regular expressions, together with the backend webserver that will deliver all URLs that match that regex. This list should be written to the file `$ZEUSHOME/balancer/mappings` and must be copied to each Balancer that is running. The mappings can be altered whilst the Balancer is still running - it will notice that the file has changed and reload the mappings as appropriate.

Cookies

The CGI or server application that provides the shopping cart can be made cluster-aware. In the HTTP response, a web server can specify that all future transactions by an individual client must be redirected to a specific backend webserver. This is achieved by the application creating a cookie named 'X-Zeus-Backend' and setting its value to the name of the backend server to be used. The next time that the Balancer receives an HTTP request by this client, it will detect this cookie. Reading the value, it will know which backend server should be used to handle the request, and forward it on appropriately. In this way, all linked transactions can be pushed onto the same backend server.

Rewriting Configuration

It is often necessary to have slightly different configuration on each web server. To make this process as easy as possible, Zeus Web Server provides a built-in mechanism for rewriting configuration settings on the fly.

When the web server receives a web site configuration from the admin server it writes it out to disk, but before it starts the server, it will look for a script called `$ZEUSHOME/web/bin/rewrite`. If this script exists, it will be run. This script takes two arguments: the first argument is the name of the virtual server, the second argument is the name of the configuration file. The rewrite program should not be group/world writeable.

This script can then rewrite the file, altering settings such as the IP bind address, document or log file locations that might vary between servers.

E.g.

```
#!/bin/sh
# $1 = vserver, $2 = file
#
# Rewrites the website name to http://`hostname`:4000/

mv $2 /tmp/z$$
sed "s/^ip_name .*/ip_name `hostname`/;s/^port .*/port 4000/" \
< /tmp/z$$ > $2
rm /tmp/z$$
```