openMosix How-to for ITS

Emma Roos
emma@fukt.bth.se

Table of Contents

1 Brief introduction

This how-to describes how to install and administrate openMosix plus how to use some of the tools that comes with it. For an introduction on this clustering system see the report.

2 Requirements

The basic hardware requirements are at least 2 network-connected computers, the faster the network the better performance you will get. The computers also have to have either Intel x86 or Athlon processors.

The system needs a basic Linux installation of any distribution though this how-to is written with RedHat in mind. The network cards needs to be properly configured and you need a quite a lot of swap-space. If you are compiling openMosix you have to have the source for a 2.4.* kernel. The openMosix tool omdiscd also needs IP multicast support enabled in the kernel.

OpenMosixview needs X, QT 2.3.0 or above, ssh (or rlogin and rsh) on all the nodes (not only the one with openMosixview on it) and the User-space tools installed.

3 Downloading

The openMosix kernel and user-space tools can be found at:
http://sourceforge.net/project/showfiles.php?group_id=46729.
Download the openMosix kernel you want (you have to have the same version on all the machines). When this how-to was written the most stable openMosix version was 2.4.19-6. If you download the rpm version don't forget to choose the one that fits best for the nodes hardware (athlon for AMD Athlon/Duron/K7 processors, i686 for Pentium-Pro/Celeron/Pentium-II/newer Pentium processors and i386 for other Intel processors or if you're not sure (and don't need SMP support). If you have more than one processor on the machine you should choose a RPM compiled with SMP support. You can check what you should use by typing *uname -a* at the prompt on the node. This will tell you what the current kernel is compiled for and if you have used the standard installation then this should be the correct kernel-type to use.
If you are going to compile openMosix download the openMosix patch with the same version number as the kernel source version you are going to compile.
Download also the User-space tools. There are two RPMs (i385 and source) and a tar.gz file with the source. Choose the version you like (the latest version that existed when this how-to was written was 0.2.4).

You will find openMosixview on *http://www.openmosixview.com/download.html.* The RPMs that exists are for RedHat 7.2 and 7.3 plus for SuSE 7.2. There is also a tar.gz with the source code. The source for QT (that is needed if openMosixview is compiled from source) can be found on *ftp://ftp.troll.no/pub/qt/source*.

4 Installation with RPMs

4.1 Introduction

The files mentioned in the installation instructions are the versions that were the latest when this how-to was written. This section only describes the installation on a RedHat system with RPMs. How to compile openMosix, the user-space tools and openMosixview is described in chapter 5 of this how-to. For information on how to install openMosix on other Linux distributions see Kris Buytaert's openMosix HOWTO.

4.2 Installing the openMosix kernel and the user-space tools

As root go to the directory where you downloaded the openMosix and user-space tools RPMs. Then use the rpm command to install the kernel and the user-space tools at the same time. For example if you are installing openMosix 2.4.19-6 on an ordinary Intel machine with one processor then type:

rpm -vih openmosix-kernel-2.4.19-openmosix6.i386.rpm openmosix-tools-0.2.4-1.i386.rpm

During this installation a script in the kernel rpm will try to add your new kernel to the boot-loader. If you have RedHat 7.2 or later with Grub as boot-loader this will work automatically otherwise you will have to add the new kernel to */etc/lilo.conf* manually.
A soft-link named */boot/vmlinuz-openmosix* always points to the new openMosix-kernel, so you don't have to edit */etc/lilo.conf* each time you upgrade the openMosix-kernel. A soft-link */boot/initrd-openmosix.img* is also created, pointing to the new openMosix-initrd. Put it in your *lilo.conf* if you need some of the modules in order to boot. Just copy the lines in */etc/lilo.conf* that lists the current kernel and change image, label and initrd.

Take and check the /etc/hosts file so that it the node's IP is listed properly in the file. If the machines IP is 192.168.0.1 and its hostname om1.itslabb.bth.se) then the host should be listed in /etc/hosts as:

192.168.0.1 om1.itslabb.bth.se
127.0.0.1 localhost

When this is done there is one of two things that should be done depending if you've decided to use the auto-discovery tool or not.

If you are not using auto-discovery then you have to edit your */etc/mosix.map* on each node. In the file each line should contain 3 fields that maps IP addresses to openMosix node-numbers:

1) The first openMosix node-number in the IP range used.
2) The IP address of the above node
3) The number of nodes in the IP range used

For example if you got 5 computers in the IP range of 192.168.0.1-192.168.0.5 the end of

/etc/mosix.map (the beginning of the file contains comments on how the file should look like) on each machine should look like this:

1 192.168.0.1 5

If you are going to use the auto-discovery daemon (omdiscd) you don't need the */etc/mosix.map*. Disable the */etc/init.d/openmosix* script with the command:

chkconfig --del openmosix

Now run *omdiscd* on each node (openMosix must be down). If you want to run omdiscd automatically when the machine reboots you have to create a startup script in */etc/init.d* (for example called *omdiscd*) and since no such script comes with omdiscd you have to create this file manually. Below is an example of how this file could look like (does only work on RedHat systems (possibly also in SuSE Linux)):

```
#!/bin/bash
#
# chkconfig: 2345 95 5
# description: omdiscd is a tool to automatically discover openMosix nodes.
#
# omdiscd           Script to stop/start omdiscd

# Source function library.
[ -f /etc/rc.d/init.d/functions ] || exit 0
. /etc/rc.d/init.d/functions

RETVAL=0

#
# The pathname substitution in daemon command assumes prefix and
# exec_prefix are same.  This is the default, unless the user requests
# otherwise.
#
case "$1" in
  start)
    echo -n "Starting omdiscd: "
       daemon /sbin/omdiscd  # default interface is eth0 if another network inteface should
                             # be used use the -i option (ex: /sbin/omdiscd   -i eth1)
    RETVAL=$?
    echo
    [ $RETVAL -eq 0 ] && touch /var/lock/subsys/omdiscd
    ;;
  stop)
    echo -n "Shutting down omdiscd: "
    killproc omdiscd
    RETVAL=$?
    [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/omdiscd
    echo
    ;;
```

```
    status)
        status omdiscd
        ;;
    restart)
        $0 stop
        $0 start
            ;;
    *)
        echo "Usage: $0 {start|stop}"
        exit 1
        ;;
esac
```

Then add this new script so that it loads at reboot by typing the command:

chkconfig --add omdiscd

Now it's time to reboot the machines, choose the openMosix kernel in the boot-loader and the cluster should work.

4.3 Installing openMosixview

First you should install openMosixview on the nodes that you want to use for monitoring the cluster. As root go to the directory where the downloaded openMosixview RPM is. Install it with the rpm command. For example if you are going to install openMosixview 1.2 for RedHat 7.3 type:

rpm -vih openMosixview-1.2-rh73.rpm

On the rest of the nodes only openMosixprocs is needed. Copy */usr/bin/openmosixprocs* from one of the machines that has openMosixview installed to */usr/bin* on all the nodes that doesn't have openMosixview installed on them.

After that openMosixview/openMosixprocs has been installed on all the nodes it's time to configure SSH so that passwords aren't needed when different features in openMosixview is used. First you have to create a RSA key-pair on all the nodes this you do with the command ssh-keygen.
If you use SSH1 you have to write *ssh-keygen -t rsa1* and if you use SSH version 2 you just change *rsa1* to *rsa*. The program will prompt you for what files you want the keys to be in (use the default) and then it will ask you for a pass phrase. You don't need a pass phrase but if you decide to have one you need to use the program *ssh-agent* before you run openMosixview.

The key pairs will be saved in two files in the */root/.ssh* directory: *identity* (private key) and *identity.pub* (public key) for SSH1 or *id_rsa* (private key) and *id_rsa.pub* (public key) for SSH2. Do **not** give out the private key to anyone!

Now copy the entire content in the public key files on the machines that openMosixview has been installed on into */root/.ssh/authorized_keys* on all the nodes in the cluster. In the end all

the nodes (both the ones with openMosixview and the ones with just openMosixprocs) should have the public keys of all the nodes with openMosixview in */root/.ssh/authorized_keys*. If the machine has openMosixview on it then its own key should also be in this file.

5 Installing from source

5.1 Introduction

The files mentioned in the installation instructions are the versions that were the latest when this how-to was written.

5.2 Compiling the openMosix kernel

You'll always have to use a pure vanilla kernel-sources to compile an openMosix kernel. The kernel source also needs to be the same version as the openMosix kernel parch you are going to use. For example if you are using openMosix 2.4.19-6 you have to use the 2.4.19 kernel.

Download the correct kernel source from *ftp://ftp.kernel.org/pub/linux/kernel/v2.4/* and unpack the kernel with: *tar -xzvf linux-2.4.19.tar.gz*

Apply the patch by going into the kernel source directory and then type the following command (if the kernel patch is in the same directory as where you downloaded the kernel source): *zcat ../openMosix-2.4.16-6.gz | patch -p1*

Now type: *make menuconfig* and enable the openMosix-options in the configuration programs. If the auto discovery daemon is going to be used then IP multicast also has to be enabled. Do also look through the other options shown and configure the kernel the way you want it.

When you exited the configuration utility its time to compile the kernel via:

make dep
make bzImage
make modules

And then install the kernel with:

make modules_install
make install

Edit your boot-loader and reboot.

There might show up problems when you try to reboot the kernel. There are some problems with certain settings in the vanilla kernel on RedHat systems. If the new openMosix kernel doesn't boot when reboot the machine with the old kernel and experiment by removing different options in the kernel. These problems are the reason why installing with RPMs are so much easier.

5.3 Compiling openMosix userland tools

Unpack the openMosix userland source: *tar -xzvf openMosixUserland-0.2.4.tgz*
Go into the directory that is created and edit the file configuration. The lines you only need to touch (most of the times) are (for paths use only **absolute** paths):

OPENMOSIX	Shows where the openMosix kernel is.
INSTALLDIR	Which is where the system base directory is (usually /).
INSTALLEXTRADIR	Where the applications are installed (usually /usr/local or /usr).
INSTALLMANDIR	Is where the man pages should be (usually $INSTALLEXTRADIR/man).
CC	What C compiler should be used (usually gcc).
MONNAME	The filename that openMosix monitoring tool should be called. The recommended name is mosmon (the name used for the program in the rpm).

When this is done it's time to compile the programs. You do this by typing: *make all*

After this copy the file *openmosix* in the scripts directory to */etc/init.d* and if you're not going to use the auto discovery tool make so that openMosix is started on reboot by the command:

chkconfig --add openmosix

If auto discovery is going to be used then the openmosix script shouldn't be added with chkconfig. Instead install the auto discovery by going into the *autodiscovery* directory that lies in the directory with the userland tools source, then remove the line "#define ALPHA" from the files *openmosix.c* and *showmap.c*. After that run the commands: *make clean* and *make*

To make the auto discovery daemon to run on reboot, create a script like the one described in the chapter 4.2 (call the file for example *omdiscd*) in */etc/init.d* and then add it so it gets started at reboot with the command:

chkconfig --add omdiscd

5.4 Compiling openMosixview

When you compile openMosixview on the machine you want to run it on you will not only need the QT libraries (that comes with the ordinary QT RPM) but you also need the QT source files available on the system. To install QT from source download the source tar.gz and then unpack it:

tar –xvzf qt-x11-free-3-0.5.tar.gz

After this you should move the entire directory created to */usr/local* with the new name *qt-x.y.z* (there x.y.z is the QT version):

mv qt-x11-free-3-0.5 /usr/local/qt-3.0.5

When this is done go into the new directory:

cd /usr/local/qt-3.0.5

Before you run the configure script you have to set the QTDIR variable so that the script find the source files.

export QTDIR="/usr/local/qt-3.0.5"

Thereafter run the configure script and compile (*make* will take quite a while):

./configure
make
make install

Now you will be able to compile openMosixview. Begin by go ing to the directory where you downloaded the openMosix tar.gz and unpack the downloaded openMosixview source code:

tar -xzvf openmosixview-1.2.tar.gz

Then go to the directory created and execute the automatic setup-script that comes with the source:

./setup [your_qt_2.3.x_installation_directory]

When openMosixview is installed copy *openmosixprocs* to */usr/bin/* on all the nodes and then fix SSH on all the nodes as described in the chapter 4.3.

6 Administrating openMosix

The openMosix cluster can be configured by using the */proc/hpc* interface and the user-space tools. OpenMosixview can also be used to configure certain settings and to monitor the cluster.

6.1 The /proc/hpc interface

In */proc/hpc* there is a number of sub directories with different openMosix settings and statistics. Some of the files in these directories can be set manually others just show statistics and is changed by the system.

The directories in */proc/hpc* are:

admin - that presents the current configuration of the system.
decay - shows the decay statistics (load balancing information settings).
info - shows information about the computer (in binary format).
nodes - has information about the different nodes in the cluster.
remote - has information about the remote processes that has migrated to the computer

There are also files with openMosix information for a specific process in the */proc/[PID*] directories (where [PID] is the process ID for the process). For more information about the different files in the sub-directories to */proc/hpc/* and */proc/[PID]* see Kris Buytaert's openMosix HOWTO.

To view the setting in any of the existing files just use cat, more, less or similar UNIX command. The only directories where the files can be manually edited is */proc/hpc/admin* and */proc/hpc/decay*. You edit these files with the echo command. For example if you want to block the machine, so it doesn't accept remote processes migrating to it, you type:

echo 1 > /proc/hpc/admin/block

6.2 User-space tools

The user-space tools have a number of commands that can be used to administrate the cluster. The commands aren't fully described here, for more information read the commands man file with *man [command]* or have a look in Kris Buytaert's openMosix HOWTO.

/etc/init.d/openmosix is the script that is used to start and stop openMosix. The options to this script are *start*, *stop* and *status*.

The command *migrate* tries to send a process to another machine. The command's syntax is:

migrate [PID] [MOSIX_ID | home | balance]

The last option can be either a nodes openMosix ID, if the process should go to the machine with the lowest load (balance) or if it should migrate to its home node (home).

The command *mosmon* is a small monitor program that displays a bar chart, normally showing the loads on the various openMosix nodes. Alternatively, it can display the various processor speeds, or the amount of available vs. total memory. The program has an online help that you can get if you type h in the program, to quit just type q. Some of the keys that can be pressed in mosmon to display different variables are:

s	shows processor speeds
m	shows memory (used out of total)
r	shows memory (raw used/free out of total)
u	shows utilizability percentage
l	go back to showing loads
d	show also dead (configured but not-responding) nodes
D	stops showing dead nodes
y	shows the yardstick in use (e.g. speed of a standard processor)

If all the nodes doesn't fit one screen you can use the left/right arrow keys to move one node to the left or right and the n or p keys to move one screen to the left or right.

The command *mosctl* is the main configuration tool. With this command you can set and read the different settings that exists in the */proc/hpc/* interface. The settings are changed by that

you type the command mosctl with different options. The syntax for mosctl is:

mosctl { stay | nostay | lstay | nolstay | block | noblock | quiet | noquiet | nomfs | mfs | expel | bring | gettune | getyard | getdecay }

When the command is used with this syntax certain settings (mainly in */proc/hpc/admin*) is set or in the case of the 3 last ones shown. The options mean:

- *stay* sets so that there won't be any automatic process migration (is cancelled with *nostay*).
- *lstay* makes so that local processes doesn't migrate to other nodes but remote processes do (is cancelled by *nolstay*).
- *block* blocks arriving guest processes (is cancelled by *noblock*).
- *quiet* disables gathering of load-balancing information (this is enabled again by *noquiet*).
- *nomfs* disables the MFS (this is enabled again by *mfs*).
- *expel* sends away all the guest processes.
- *bring* brings all the migrated processes home
- *gettune* shows the current overhead parameters used by the kernel to estimate the "I/O factor" in its load-balancing.
- *getyard* shows the current yardstick (the processor speed of the most typical openMosix node).
- *getdecay* shows the current decay parameter (controls the gradual decay of old process statistics for the use of load-balancing).

mosctl whois [OpenMosix-ID | IP-address | hostname]

Resolves openMosix ID, IP-addresses and hostnames of the cluster.

mosctl { getload | getspeed | status | isup | getmem | getfree | getutil } [OpenMosix-ID]

Displays different status parameters on a certain node (defined by its openMosix ID).

- *getload* displays the current (openMosix-) load
- *getspeed* displays the current (openMosix-) speed.
- *status* displays the current status and configuration (stay, lstay, block, quiet).
- *isup* tells if a node is up or down.
- *getmem* shows the logical free memory.
- *getfree* shows physical free memory.
- *getutil* displays the utilization.

mosctl setyard [processor-type | number | this]

Sets a new yardstick value (can be good if the majority of the nodes are very different from the standard yardstick).

mosctl setspeed [numeric-value]

Overrides the node's idea of its own speed.

mosctl setdecay [interval] [slow] [fast]

Sets a new decay interval: *interval* in seconds, how much of 1000 to keep for *slow*-decaying processes and how much of 1000 to keep for *fast*-decaying processes. The interval must be within the range of 1-65535 and the slow and fast parameters in the range of 1-1000 (there the slow parameter is greater or equal to the fast parameter).

For more information on the different settings read mosctl's man page.

The command *mosrun* is used to execute jobs on the cluster with special openMosix settings on a specific node (or nodes). There are also several scripts that can be used to execute a job with a special pre-configured openMosix configuration. The scripts are: *nomig*, *runhome*, *runon*, *cpujob*, *iojob*, *nodecay*, *slowdecay* and *fastdecay*. Mosrun isn't needed to run processes that uses the openMosix migration but is mainly there if you want to control how the process you are starting should behave.

The command *setpe* is used to configure what nodes are in the cluster by reading a specified file (*setpe -w -f [filename]*). You can also read the current configuration with the program (*setpe -r*) plus shutting down openMosix by removing the configuration (*setpe -off*).

The commands *mtop* and *mps* are openMosix aware versions of the standard Linux commands *top* and *ps*. These commands have an extra column that shows what node the programs are running on (in a column marked *n#*). Only the programs that started on the node that mtop/mps is running on is shown and programs that hasn't migrated to another node has node number 0. The options used by these two programs are the same as the ones used for ordinary ps/top.

The tool *omdiscd* is an auto discovery daemon that can be used instead of */etc/mosix.map*. If you start omdiscd openMosix has to be shutdown:

/etc/init.d/openmosix stop

Then you start the auto discover daemon with the command *omdiscd*. If the machine has more than one network interface the -i option should be used. You can also start the daemon so it runs in the foreground sending messages and debugging output to standard error. The option is -n and without it all output should go to the syslog.
For example if you only want omdiscd to run in the foreground and also to search for openMosix nodes on the network that the eth1 interface is connected too then type:

omdiscd -n -i eth1

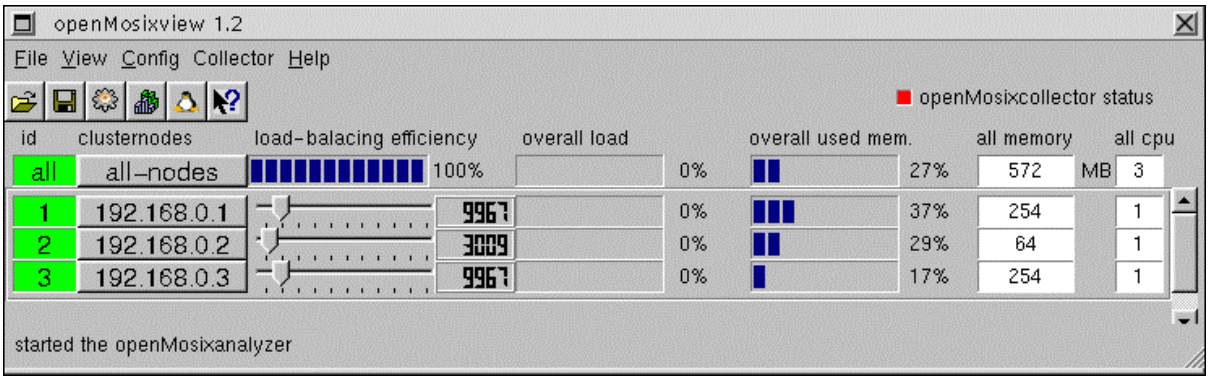
The auto discovery daemon comes with another command called *showmap*. This command shows the current nodes in the cluster.

6.3 openMosixview

The tool openMosixview is a graphical interface for doing certain administration and monitoring tasks on an openMosix cluster. OpenMosixview consists of five programs there among a main program (called just openMosixview) that is used to access the other four. When you run the command *openmosixview* do this as root (so you can change the different

settings) and don't start the program in the background since it wants to use the console that the program started from for error messages etc.

When openMosixview has started you will get a window that shows the nodes on your cluster and their status.



The status information is shown in several columns, with the top row showing the entire clusters status, these columns are:

- **Id:** The node number with a background color that shows if the node is up (green) or down (red).
- **Clusternodes:** A button with the nodes IP-address on, if the button is pressed a window shows up. This new window can be used to set some of the nodes settings (the window that shows up if the button *all-nodes* are pushed sets these settings for all the nodes). In this window you can:
  - Turn auto migration on and off.
  - Tell the node if it should talk to other nodes.
  - Set the node so local processes should stay on it.
  - Tell the nodes guest processes to leave the node.
  - Start and stop openMosix.
  - Open a console on the node.
  - Open the program openMosixprocs on the node.
  - Tell openMosixview on what machine the console and the openMosixprocs program should be displayed on (preferably the machine where you sit at the monitor and run X windows).
- **Load-balancing efficiency:** On the first row where the entire cluster information is shown displays how effective the load balancing is. On the rows that show information about specific nodes a slider exists that shows the current speed of the node and if the slider is moved this speed is changed. Next to the slider is the numerical value of the speed.
- **Overall load:** The load on the entire cluster and the different nodes.
- **Overall used mem:** The overall used memory on the entire cluster and the different nodes.
- **All memory:** The physical memory on the entire cluster and the specific nodes in megabytes.
- **All CPU:** The first row shows the number of CPU's on the entire cluster. The other rows show the number of CPU's on a specific node.

There are five drop-menus at the top of the program these are:

- **File:** In this menu you can either choose to quit openMosixview (Exit) or to get up the advanced execution window (run programm). If you choose the advanced execution window you'll first get to choose what program you want to run on the cluster. When the program has been chosen you'll get a window where you can set the options that the program should have and how you want it to run on the cluster (for example if you want the program to run on a specific node). These settings are similar to the ones that can be set by the program mosrun. When you have set the settings you should press the execute button that's in the window and the program will start.
- **View:** There you can set if you want a status bar at the bottom of the main window that shows what feature was last executed in openMosixview (statusbar).
- **Config:** Here you can set if you want to use SSH when openMosixview talks to the different nodes (use ssh). Here you can also save the configuration (save conf).
- **Collector:** Here you control the openMosixcollector. In this menu is a submenus (openMosixcollector) there you can *start*, *stop*, put in a *checkpoint* and save the openMosix history (*save history*). The status of the openMosixcollector is shown by a little status display (green for on and red for off) in the main window.
- **Help:** In this menu you can get up a window with an online help (*Help*), a mouse pointer that you can use to click on different parts of the windows to get a description what part in the program does (*What is this*) and to get a window with information about the current version of openMosixview (*About*).

In the main window there is also six buttons that represents different features in the program these are:

 This button has the same feature as the menu choice: File -> Run programm.

 This button saves the rsh/ssh configuration (same as Config -> save conf)

 Opens up openMosixprocs on the node that openMosixview is running on.
In openMosixprocs there is a list of all the processes on the current node (with a dropdown box to choose if you want to list a specific user's processes). If you double click on a listed process a window with information about the process is shown plus that you can tell the process to migrate, die and some other things. In the main openMosixprocs window there is also a button that brings up a window there you can manage the remote processes that have migrated to the node. In this window there is a tab for each remote process with information about the process and two buttons, one for migrating the process to its home node and one to migrate the process to the node with the lowest load.

 Opens the program openMosixanalyzer that shows a graphical display of the data collected with openMosixcollector.
In openMosixanalyser you can view a graph over the load () and memory () during the time openMosixcollector has been on by clicking the two icons to choose each graph. The window shows one graph per node in the cluster plus one for the entire cluster. Next to each graph are two icons one that shows the minimum, maximum and average load and memory () and one for printing out the graph on a printer (). A third icon () next to the ones that pull up the load and memory graphs starts up the program openMosixHistory that shows the processes on the node during certain times (chosen by

a slide bar on the top of the window).

 This button has the same feature as the menu choice: Help -> Help.

 This button has the same feature as the menu choice: Help -> What is this.

7 Links to more information

The official web page for openMosix: *http://www.openmosix.org*
The official web page for openMosixview: *http://www.openmosixview.com*
Kris Buytaert's openMosix HOWTO: *http://howto.ipng.be/openMosix-HOWTO*
OpenMosix Documentation Wiki: *http://howto.ipng.be/openMosixWiki/*