Storage (Domain) Threshold Alerts

Design Specification

Version 0.3

NW

CASL

Building ZK3
110 Spit Brook Road
Nashua, NH 03062

Primary reviewers:

Design Specification Revision History		
Version	**Date**	**Changes**
0.1	01/12/04	First draft for internal review
0.2	01/14/04	Addressed comments from review
0.3	01/29/04	Addressed DS PRT comments

1. Introduction

1.1 Overview

This project is intended to provide a means for administrators and/or the system itself to respond to conditions of increased or decreased consumption beyond a predetermined (threshold) limit of a filesystem's available storage pool.

On Tru64 Unix via the AdvFS GUI, each AdvFS Domain and fileset could optionally have a "free space alert" set. When free space used by a given object crossed the alert threshold (based upon percentage in use) an alert was issued. Upon issuance of an alert, a shell script may have optionally been executed.

1.2 Impacts

This project implements threshold management via AdvFS utilities and threshold monitoring by the kernel with alerts generated in the form of EVM events.

Due to the fact that on HPUX the user is not presented with the concept of fileset domains, even though "under the covers" the domain/fileset design still exists, AdvFS on HPUX will present thresholds as Storage thresholds. A Storage threshold to the user is going to be implemented as a domain threshold in the code. They will be settable on original (non-snapset) Storages only, although any storage consumed by a snapset will come from its parent filesystem's storage pool and ultimately contribute to hitting its Storages thresholds.

The original Tru64 Unix design implemented threshold functionality for filesets as well. Although initially we will only be exposing domain thresholds to the user (as Storage thresholds), the supporting code for fileset thresholds will be defined in this document and implemented in the coding stage as well.

As we still utilize a domain/fileset design in our AdvFS code, we will be implementing kernel APIs, data structures, transaction FTX agent IDs and RootDone records whose naming conventions follow the domain/fileset maxim.

2. Proposed Design

2.1 High Level Design Overview

In this initial version of the threshold project, Storage(domain) thresholds will be exposed to the user. Fileset thresholds will be implemented as well, but not presented to the user via any API. Possibly sometime in future versions, fileset thresholds will be made available.

A threshold can be defined as follows:

A threshold consists of a limit, interval, and an elapsed time value. The limit and interval are values set by the user. The kernel is responsible for managing the elapse time value.

The limit value is an integer that represents percent full for the Storage. For a given filesystem(domain) or fileset there will be the potential for setting two threshold limits. There can be an upper limit threshold set and a lower limit threshold set. These can be thought of as high and low water marks.

The interval value is an integer that equals the number of minutes that must elapse prior to generating subsequent EVM events. This is to eliminate the possibility of a flood of EVM events being generated as storage continues to be allocated or deallocated around the threshold limit. It is important to note two things. First is that events are generated only when a request for storage takes us from being below a threshold limit to being above that threshold limit, and that future storage allocation above the limit would not generate an event regardless of the event interval. Second is that users can set the interval value to zero. This relinquishes ability of AdvFS to control a potential flood of EVM events should storage thrash around the threshold limit.

The proposed design is divided into three parts. The first deals with displaying threshold values, the second deals with managing threshold values, and the third deals with monitoring for threshold crossing and generating EVM events when conditions have been satisfied.

2.1.1 Displaying Threshold values

Storage threshold values can be viewed by users via the AdvFS utility fsadm_advfs from the fsadm wrapper command. The following goes through the high level design for viewing Storage thresholds. Remember that these are the equivalent of Domain thresholds.

2.1.1.1 User interface

Storage threshold values can be viewed via the fsadm info command which will access the kernel via the existing API msfs_get_dmnname_params().

Fileset threshold values will be retrievable using the existing API msfs_fset_get_info() but will not be utilized in the initial version of this project.

2.1.1.2 Kernel interface and syscall implementation

The libmsfs.so library interface makes a kernel syscall with the op-type ADVFS_OP_GET_DMNNAME_PARAMS.

msfs_syscall_op_get_dmnname_params() will be the routine that gets called for the operation. The threshold limit, event interval and time of last event are copied into the user's buffer from the in-memory domainT structure.

2.1.2 Managing Threshold Values

User settable threshold values consist of both a threshold limit and an event interval. A valid limit will be between 0 and 100, corresponding to "percent full". A valid event interval will be between 0 and 10080, corresponding to "minutes between EVM events". Intervals will be settable via a minute, day. Some examples would be 5m or 3d. All would be converted to minutes and dealt with by the kernel as such. Thresholds are initialized at domain and fileset creation time to a limit of 0. They remain inactive until activated by a System Administrator by changing either the upper or lower limit to a valid non-zero value. The following walks through the high level design for managing each.

2.1.2.1 User interface

Storage thresholds will be managed via the fsadm –chfs utility and will access the kernel via the new API advfs_set_dmnname_params ().

The caller will have the ability to activate or deactivate threshold monitoring for a given named, non-snap filesystem. If the caller wants to activate threshold monitoring a valid filesystem name and threshold limit must be given. This can be the upper and/or lower limit. Providing an event interval is optional as a default value of 5 minutes will be used if omitted by the caller. If the caller wishes to deactivate threshold monitoring for a filesystem, fsadm –chfs would be called with a valid Storage name and a threshold limit of 0 for both the upper and lower values.

2.1.2.2 Kernel interface and syscall implementation

Storage (Domain) thresholds:

The libadvfs library interface makes a kernel syscall with the new op-type ADVFS_OP_SET_DMNNAME_PARAMS.
advfs_syscall_op_set_dmnname_params() is the handling routine that gets called for the operation.

Finally, bs_set_dmn_threshold_info() is called and the domain threshold limits and event interval are copied into the in-memory domainT and written out to disk in the BSR_DMN_MATTR record of the RBMT. This record resides in the RBMT of the volume that holds the most current Log for the domain.

Fileset thresholds:

The libadvfs library interface makes a kernel syscall with the op-type ADVFS_OP_SET_BFSET_PARAMS. advfs_syscall_op_set_bfset_params() is the handling routine that gets called for the operation.

Finally, bs_set_bfset_threshold_info() is called and the fileset threshold limits and event interval are copied into the in-memory bfSetT and written out to disk in the BSR_BFS_ATTR record for the fileset. This record resides in the RBMT of the volume that holds the most current Log for the domain.

2.1.3 Threshold monitoring / EVM event generation

2.1.3.1 Upper Limit

Storage(domain) and fileset threshold upper limits are checked anytime additional on-disk storage is requested in bs_stg.c:add_stg(). When new storage is requested, if the upper threshold limit would be crossed by the request, and the event interval has been exceeded as compared to the time of the last EVM threshold event, a rootdone operation is attached to the parent transaction handling the request for additional storage. When the parent transaction completes, the rootdone operation is executed and an EVM event is generated announcing the crossing of the threshold limit. The rootdone approach guarantees that an EVM event will not be generated unless the request for additional storage succeeds.

2.1.3.2 Lower limit

Storage(domain) and fileset threshold lower limits are checked anytime on-disk storage is removed by file deletion in bs_access.c:bs_close_one() and when files are truncated in bs_stg.c:dealloc_stg(). When storage is being removed, if the lower threshold limit would be crossed by the request, and the event interval has been exceeded as compared to the time of the last EVM threshold event, a rootdone operation is attached to the parent transaction handling the storage deallocation. When the parent transaction completes, the rootdone operation is executed and an EVM event is generated announcing the crossing of the threshold limit. The rootdone approach guarantees that an EVM event will not be generated unless the storage deallocation succeeds.

3. Detailed Design Overview

The following describes the threshold project design in greater detail. The approach will be to define new and changed data structures, EVM events, FTX transaction types, and new functions for the three general subsystems of the project followed by pseudo code of the logic for each of these subsystems. Again, these are threshold viewing, threshold management, and threshold monitoring. For the sake of simplicity, threshold management and threshold viewing will be combined in one subsection. First are listed the new and changed data structures common to all three subsystems.

3.1 New or Changed Structures for Storage(domain)/fileset Thresholds

3.1.1 bs_public.h:adv_threshold_t and adv_threshold_ods_t

These new data structure describe thresholds for domains and for filesets. This structures will be incorporated into existing in-memory structures in the case of adv_threshold_t and on disk in the case of adv_threhsold_ods_t.

```
/*
 * holds domain and fileset threshold values in memory.
 */
typedef struct {
    uint32_t threshold_upper_limit;
    uint32_t threshold_lower_limit;
    uint32_t upper_event_interval;
    uint32_t lower_event_interval;
    uint64_t time_of_last_upper_event;
    uint64_t time_of_last_lower_event;
} adv_threshold_t;


/*
 * holds domain and fileset threshold values on disk.
 */
typedef struct {
    uint32_t threshold_upper_limit;
    uint32_t threshold_lower_limit;
    uint32_t upper_event_interval;
    uint32_t lower_event_interval;
} adv_threshold_ods_t;
```

3.1.2 bs_ods.h:bsDmnMAttrT

This is the existing on-disk RBMT record defined to hold domain mutable attributes with the new thresholdT structure added.

```
#define BSR_DMN_MATTER 15
typedef struct {
    onDiskVersionT ODSVersion;
```

```
    .
    .
    .
    thresholdT     threshold;
    .
    .
} bsDmnMAttrT;
```

3.1.3 bs_ods.h:bsBfSetAttrT

This is the existing on-disk RBMT record defined to hold fileset attributes with the new thresholdT structure added.

```
#define BSR_BFS_ATTR 8
typedef struct {
    bfSetIdT bfSetId;
    .
    .
    .
    thresholdT     threshold;
    .
    .
} bsBfSetAttrT;
```

3.1.4 advfs_evm.h: #defines

These are the new EVM event types needed to support this project. Note the use of STORAGE in the naming conventions due to exposure to the user.

```
#define EVENT_STORAGE_UPPER_THRESHOLD_CROSSED
_EvmSYSTEM_EVENT_NAME("fs.advfs.storage.upper.threshold.crossed")

#define EVENT_STORAGE_LOWER_THRESHOLD_CROSSED
_EvmSYSTEM_EVENT_NAME("fs.advfs.storage.lower.threshold.crossed")

#define EVENT_FSET_UPPER_THRESHOLD_CROSSED
_EvmSYSTEM_EVENT_NAME("fs.advfs.fset.upper.threshold.crossed")

#define EVENT_FSET_LOWER_THRESHOLD_CROSSED
_EvmSYSTEM_EVENT_NAME("fs.advfs.fset.lower.threshold.crossed")
```

3.1.5 advfs_evm.h:advfs_ev

This is the modified advfs_ev structure. It's used to pass data values to the EVM. The only change is the addition of threshold_limit values and event_interval values.

```
typedef struct _advfs_event {
       char    *special;
       char    *domain;
       .
       .
       .
       uint32_t threshold_upper_limit;    /* NEW */
       uint32_t threshold_upper_interval; /* NEW */
       uint32_t threshold_lower_limit;    /* NEW */
       uint32_t threshold_lower_interval; /* NEW */
} advfs_ev_t;
```

3.2 New or Changed Structures and Functions for Threshold Viewing / Management

3.2.1 msfs_syscalls.h:ml_threshold_t

This is a new structure that is used as the buffer for passing threshold data between the user commands and the kernel via the AdvFS system calls for viewing and managing thresholds.

The freeBlks and totalBlks contain data needed at the user level after a threshold has been set with the fsadm_advfs utility. The utilities make an initial check for threshold crossing and need these fields to accomplish this.

```
typedef struct {
    thresholdT threshold;     /*  dmn or bfSet threshold values      */
    uint64_t freeBlks;        /*  total free blocks in dmn or bfSet  */
    uint64_t totalBlks;       /*  total blocks in dmn or bfSet       */
    uint64_t unused[10];      /*  zeroed and reserved for future use */
} ml_threshold_t;
```

3.2.2 msfs_syscalls.h:mlDmnParamsT

This is the new mlDmnParamsT structure that now includes an instance of an mlThresholdT.

```
typedef struct {
    mlBfDomainIdT bfDomainId;  /* domain id        */
    .
    .
    .
    mlThresholdT threshold;    /* domain threshold */
} mlDmnParamsT;
```

3.2.3 msfs_syscalls.h:libParamsT

The following are the one new and three existing union members of the libParamsT structure used when making the four AdvFS system calls for threshold viewing/management. The structure name infers the system call it is associated with.

New:

```
typedef struct {
    char        *domain;           /* in   - domain name   */
    mlDmnParmsT  dmnParms;         /* in   - domain params */
} opSetDmnNameParamsT;
```

Existing:

```
typedef struct {
    char        *domain;           /* in   - domain name   */
    mlDmnParmsT  dmnParms;         /* out  - domain params */
} opGetDmnNameParamsT;
```

```
typedef struct {
   mlBfSetIdT     bfSetId;         /* in  - fileset id     */
   mlBfSetParamsT bfSetParams;     /* in  - fileset params */
} opSetBfSetParamsT;
```

```
typedef struct {
   mlBfSetIdT     bfSetId;         /* in  - fileset id     */
   mlBfSetParamsT bfSetParams;     /* out - fileset params */
} opGetBfSetParamsT;
```

3.2.4 bs_params.c:bs_set_dmn_threshold_info()

This is a new routine defined to set domain threshold values in memory and on disk as a result of an instance of the fsadm -chfs utility.

definition:

```
statusT
bs_set_dmn_threshold_info(
    domainT    *dmnP,          /* in - domain pointer   */
    thresholdT *threshold      /* in - threshold pointer */
)
```

logic:

```
check validity of domain pointer.
start a root transaction.
    if fail then return failure value.
get domain's log vd pointer.
malloc memory for a bsDmnMAttrT structure buffer.
    if fail then end transaction and return ENO_MORE_MEMORY.
copy threshold data into malloc'ed memory buffer.
write buffer to BSR_DMN_MATTR record on vd of domain log.
    if fail then free memory buffer, end ftx and return failure value.
copy threshold data to domainT in memory.
free malloc'ed memory buffer.
end transaction.
return EOK.
```

3.2.5 bs_bitfile_sets.c:bs_set_bfset_threshold_info()

This is a new routine defined to set fileset threshold values in memory and on disk as part of future functionality. A user interface needs to be defined as part of that future work.

definition:

```
statusT
bs_set_bfset_threshold_info(
    bfSetT     *bfSetp,       /* in - bitfile-set's desc pointer     */
    thresholdT *threshold     /* in - bitfile-set's threshold struct */
)
```

logic:

```
check validity of bfset pointer.
start a root transaction.
    if fail then return failure value.
malloc memory for a bsBfSetThrshldAttrT structure buffer.
    if fail then end transaction and return ENO_MORE_MEMORY.
copy threshold data into malloc'ed memory buffer.
write buffer to BSR_BFSET_THRESHOLD_ATTR record into m-cell chain of
filesets Tag file.
    if fail then free memory buffer, end ftx and return failure value.
copy threshold data to bfSetT in memory.
free malloc'ed memory buffer.
end transaction.
return EOK.
```

3.3 New or changed Data structures and Functions for Threshold Monitoring

3.3.1 ftx_agents.h:ftxAgentIdT

A new transaction agent will be added to handle generating threshold EVM events. The new agent added to the enumeration is FTA_ADVFS_THRESHOLD_EVENT.

3.3.2 bs_stg.c:thresholdAlertRtDnRecT

This is the record for the FTA_ADVFS_THRESHOLD_EVENT agent that is attached to the transaction passed to add_stg() and dealloc_stg(). It is passed to the rootdone routine upon completion of the root transaction. First there is a threshold type enumeration to denote domain or fileset threshold followed by the definition of the thresholdAlertRtDnRecT.

```
typedef enum {
    DOMAIN_UPPER = 0,
    DOMAIN_LOWER = 1,
    FSET_UPPER   = 2,
    FSET_LOWER   = 3
}thresholdTypeT;
```

```
typedef struct thresholdAlertRtDnRec
{
    thresholdTypeT type;
    bfSetIdT       bfSetID;
}thresholdAlertRtDnRecT;
```

3.3.3 bs_stg.c:init_bs_stg_opx()

Add a call to ftx_register_agent() registering advfs_threshold_rtdn_opx() as the rootdone routine for an FTA_ADVFS_THRESHOLD_EVENT agent.

```
sts = ftx_register_agent(
                         FTA_ADVFS_THRESHOLD_EVENT,
                         NULL,                        /* undo      */
                         &advfs_threshold_rtdn_opx    /* root done */
                        );
```

3.3.4 bs_stg.c changes to add / remove storage routines

3.3.4.1 add_stg()

Additions to this routine are made to facilitate the monitoring of domain and fileset upper limit thresholds. This code will be added to the beginning of the routine.

logic:

Calculate potential domain and fileset blocks used if the request for storage was to succeed.

```
if domain upper threshold limit is > zero and
   total domain blocks is < the upper threshold limit and
   potential domain blocks is > upper threshold limit and
   event interval has expired then
       Start a sub transaction and fill in rootdone record.
       Attach the FTA_ADVFS_THRESHOLD_EVENT agent.
       Set rootdone record type to DOMAIN_UPPER.
       End sub transaction.

if fileset upper threshold limit is > zero and
   total fileset blocks is < the upper threshold limit and
   potential fileset blocks used is > upper threshold limit and
   event interval has expired then
       Start a sub transaction and fill in rootdone record.
       Attach the FTA_ADVFS_THRESHOLD_EVENT agent.
       Set rootdone record type to FSET_UPPER.
       Set the bfSetId in the rootdone record equal to the
       fileset's bfSetId.
       End sub transaction.
```

3.3.4.2 dealloc_stg() and bs_close_one()

Additions to this routine are made to facilitate the monitoring of domain and fileset lower limit thresholds. This code will be added to the beginning of the routine.

logic:

Calculate potential domain and fileset blocks used if the deallocation of storage was to succeed.

```
if domain lower threshold limit is > zero and
   total domain blocks is > threshold limit and
   potential domain blocks is < threshold limit and
   event interval has expired then
       Start a sub transaction and fill in rootdone record.
       Attach the FTA_ADVFS_THRESHOLD_EVENT agent.
       Set rootdone record type to DOMAIN_LOWER.
       End sub transaction.

if fileset lower threshold limit is > zero and
   total fileset blocks is > threshold limit and
   potential fileset blocks is < threshold limit and
   event interval has expired then
       Start a sub transaction and fill in rootdone record.
       Attach the FTA_ADVFS_THRESHOLD_EVENT agent.
       Set rootdone record type to FSET_LOWER.
       Set the bfSetId in the rootdone record equal to the
       fileset's bfSetId.
       End sub transaction.
```

3.3.5 bs_stg.c:advfs_threshold_rtdn_opx()

This is the rootdone operation associated with the FTA_ADVFS_THRESHOLD_EVENT agent. This is the routine that generates an EVM event when a domain or fileset threshold has been crossed. Using the rootdone approach ensures that an event is not prematurely generated if the addition or deallocation of storage were to fail.

definition:

```
void
advfs_threshold_rtdn_opx(ftxHT  ftxH,  /* handle to transaction      */
                         int32_t size,  /* size of rootdone record    */
                         void* address  /* address of rootdone record */
                        )
```

logic:

```
copy rootdone record from address to local thresholdAlertRtDnRecT.
get system time.

if rootdone record type is DOMAIN_UPPER then
    generate an EVENT_STORAGE_THRESHOLD_UPPER_CROSSED EVM event
    update domain->threshold.time_of_last_upper_event with system time
    return.

If rootdone record type is DOMAIN_LOWER then
    generate an EVENT_STORAGE_THRESHOLD_LOWER_CROSSED EVM event
    update domain->threshold.time_of_last_lower_event with system time
    return.

if rootdone record type is FSET_UPPER or FSET_LOWER then
    if domain of the transaction is not active then open fileset.
        if open fails then panic domain and return.

        if rootdone record type is FSET_UPPER then
            generate an EVENT_FSET_THRESHOLD_UPPER_CROSSED EVM event.
            update bfSetP-> threshold.time_of_last_upper_event with
            system time.
        else if rootdone record type is FSET_LOWER then
            gerneate an EVENT_FSET_THRESHOLD_LOWER_CROSSED EVM event.
            update bfSetP-> threshold.time_of_last_lower_event with
            system time.

if fileset was opened then close.
    if fail then panic domain.

return.
```

Appendix

I. Possible Future Enhancements

1. Implement fileset threshold viewing and management if the domain/fileset environment is exposed to the user in a future HPUX release.

2. Add initial checks for threshold crossing at domain activation and fileset mount time.

3. Add rootdone operation to parent transaction passed to add_stg() directly, saving having to start a sub transaction to attach the FTA_ADVFS_THRESHOLD_EVENT agent.