Customer BIOS Configuration Challenges and Solution

John Wang, Intel Inc.

Abstract

Software configuration is not only a verification challenge, but also a threat to system functionality and stability. The post analysis of the customers' critical issues shows the Intel BIOS default configuration is modified by some customers. To block the issues from being leakage to customers, this paper discusses some topics being impacted by the BIOS configuration customization. From the software engineering viewpoint, we discussed the root cause and challenges of the customer's modification of the BIOS, proposes improvement suggestions. For the existing issues, 2 workflow and toolchains were developed to get a full picture of the customer change list. Based on the information an aggressive strategy is developed to check out the potential issues before release. The regressions show more than half of the issues that were previously thought may be leaked to customers can be discovered before the BIOS release.

Keywords: requirement, BIOS, configuration, change, issue, aggressive, validation

1 Introduction

In the past years, some Intel partners discovered some critical or high exposure issues on their platforms, which have been classified as the Intel BIOS reference code related. The platform validation team joined the debug task forces, paid some efforts to support the customer issue debugging, try to understand the customer demands, designed some use cases to simulate the customer scenarios to reproduce the customer issues on the Intel reference platforms.

During the debugging we found a common phenomenon that we must follow the customers to change some system settings, otherwise the issue cannot be reproduced on the reference platform. The changes maybe not locate in the BIOS directly, but almost all changes corresponding to an exposed or unexposed BIOS option.

For example, we can disable mwait by Linux kernel parameter "idle=nomwait" or disable the "Enable Monitor MWAIT" option in the BIOS setting interface, these 2 kinds of operation will get the same result. The new designed test cases also proved that we could use either of these methods to change the system settings. Because most of the customer configuration changes locate in the BIOS, we prefer to centrally control the modifications in the BIOS either.

The post analysis shows a common property that almost all such kind of issues are related with the BIOS setting modification, and the big statistics data shows the customers come from the same business field will do the same change other than the Intel default setting. This fact brings us some challenges:

1. Why the customers modify the BIOS default settings? Why do customers from the same field who have a competitive relationship use the same setting for the same option?
2. Why the BIOS option changing will cause issue?
3. How to find the difference? What options did they change?
4. Can we block the exposed holes in the future projects?
5. How to avoid or reduce such risk in the future?

Configuration management is more important than ever because customers want new production designs of higher quality at lower prices. The complex customer segment improved the complexity.

For the existing hole, a problem is even most of the customers themselves do not have a clear document to manage their change requirements and results. we developed 2 tool chains and workflows to collect the information, applied an engineering strategy to weaken or even fix their effect.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

2 Body

2.1 Root cause

Why the customers modify the BIOS default settings? Beside the BIOS configuration modification, another behavior was discovered when comparing the modified prefers among different customers, the customers from the same field will use the same setting for the same option, even they have a competitive relationship. We can explain this phenomenon by software requirement management.

Intel wants to fulfill all the user requirements. In an ideal environment, we satisfy almost all the requirements and release them in time. However, the actual environments are quite different. Tight budget and schedule imply some serious requirement issues and challenges. One of the most critical issues is unvoiced customer belief i.e. they wasted their valuable time in specifying requirements for a feature we have strong business judgement, for example, a feature which has been moved into ZBB (Zero Based Budgeting), or a POR (Plan of Record) feature that must be enabled by default, like the RAID (Redundant Array of Inexpensive/Independent Disks) disk management.

When the customer cannot be satisfied directly by Intel, they will go to another independent BIOS vendor. Although the BIOS reference code still comes from Intel, the dedicated vendor will do some customization for the customer. In most cases, the vendor will have several similar customers come from a common business field, they will have similar requirements for a common feature, their advocation for the setting may propagate to the other customers in that field.

2.2 Why the BIOS option changing will cause issue?

A scenario is created to illustrate the BIOS configuration process limitations. A BIOS code base contains multiple options from F0 to Fn and constrains from C0 to Cm. Two derivate BIOS B1 & B2 configure from the BIOS code base.

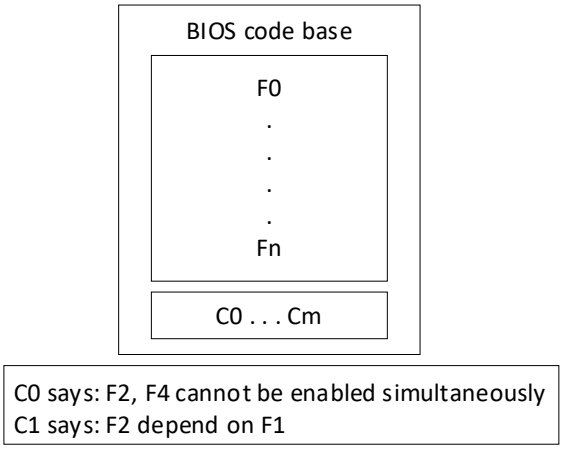

Figure 1: BIOS Repository

Figure 2 describes the configuration scenario with additional requirements. Configuration of B1 starts first. Option F1 and F2 configure respectively. When configuring option F3, customer gives some additional requirements that are not supported by the current BIOS branch and these requirements are mandatory for customer business process. Traditional configuration management process focuses on selection and assembling of components from existing branch and do not support the software requirement engineering, so every configuration manager deals this situation in his own way without going in detail. Such a weak, substandard, undefined requirement gathering process can introduce critical problems in later phases and the complexity of configuration will be increased. Lack of a well-defined new feature requirement places a big question mark on production quality, it will impact the user satisfaction and the ability of future improvement to entertain additional user wishes. This may in a great chance that the user refuse to accept the product. No support for the software requirement engineering is a main cause.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

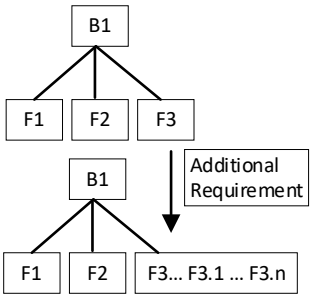

Figure 2: Configure with additional requirements

Figure 3 illustrates another common scenario. The 2nd derivative BIOS B2 configures with the option F2, F3, F4 and F7. The testing will check out that B2 has violated the constrain C0, i.e. F2 and F4 cannot be enabled simultaneously, and C1, i.e. F2 enabled but its dependent F1 is not enabled.

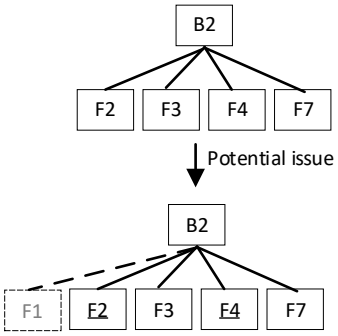

Figure 3: Configure conflictions

Late discovery of the constrain violation introduces complexity in configuration process and moves B2 in an inconsistent state. Transform from a flawed featured model into another constraint satisfying model is complex, resolve the conflicts during the transformation is a time consumption process.

Intel have a good production requirement management engineering practice, may co-work with the IBVs to have a better understand and handle for our customer requirements.

2.3 How to find the difference? What options did they change?

The number of server platform BIOS configuration options exceeds 8000, new platforms will add more options that the total number will exceed 10000. It's impossible to manually check them.

Another difficulty is the option location. Different developers may name the same option with different names, it's a serious challenge for the software program of automation analysis.

Sometimes, the crashed platform may not follow the default settings, it may have some interim parameters under tuning that even the operator does not record them. When a crash happen, the operator does not sure the exact settings.

We have developed some tools to do the analysis from different path. One path is analyzing the source design files, as the figure 4 shows. The other path is to fetch the binary information from the crashed platform with the system runtime configuration, will be described later in this paper.

We got the customers support for the post analysis. They provided their configuration files, although the files are in some very different formats; we follow the data process flow shows in figure 4 below, convert different data formats into an identical data structure then statistic the difference.

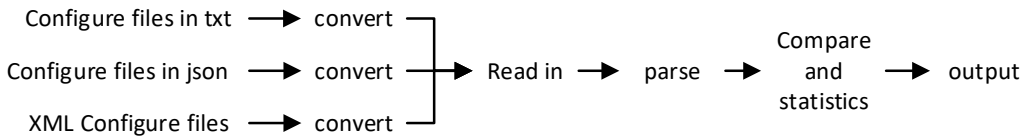

Figure 4: Get customer BIOS configuration change list from design files

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

For platforms with complete configuration files and BIOS setting change records, we can get the list follow the workflow shows in figure 4. For a platform that presents the scene of the accident, maybe we don't know which configuration file it uses, or we are not sure what the exact settings are, so we have developed a new workflow to obtain the difference between the current configuration of this device and the configuration of the reference platform by comparing the register contents. The new workflow is shown as Figure 5.

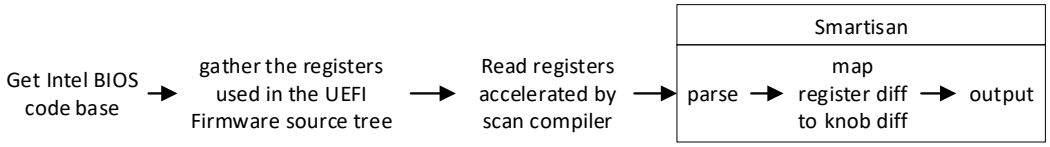

Figure 5: Get customer BIOS configuration change list from registers contents

The changed options can be classified into 3 categories.

1. Customers don't use some features, so the corresponding settings are turned off.
   Usually such options are exposed for new features, the customers do not enable the new features before their evaluation passed. By default, these options are enabled in the Intel BIOS reference code, with them be enabled the platform can expose more issues that help us fix the developing code issues earlier.
2. Some customer settings do have better chance to expose issues, e.g. IIO error pin.
   The customers wish the hardware issues are visible, so they enabled the hardware interrupt pin.
3. Customer may disable some PnP (Power and Performance) related features.
   The platform power and performance are impacted by these changes. This is a category that may cause some power or performance issues, so if possible, the validation team will add coverage for the customer settings.

2.4 Can we block the exposed holes in the future projects?

This question can be asked another way: can we reproduce the known customer issues with the reference platform? If we can detect the issue in the inner validation, it will not leakage to the customers anymore.

Reproduce the customer issue is always another challenge for system validation, especially when we have requirements for the reproduce speed and complexity. Usually the customer environment has a different hardware infrastructure includes the mainboard and peripherals, the BIOS configuration is therefore somewhat adaptable to the hardware, all these makes the customer application runs in a totally different environment than the Intel reference platform.

Fortunately, if the issue happened on the customer platform is really an escaped bug in the developing code, we can use a well-designed case to reproduce it, although sometimes we need a brainstorm to develop a case that looks completely different to meet all the reproducing conditions. Sometimes it takes a bit of luck to discover changes that the customers themselves didn't even realize.

Use a real case as the sample. A customer has a segment fault issue when they incidentally changed some default BIOS settings, including they disabled the mwait instruction with a Linux kernel parameter, and run a corner multi-threading case.

It's easy to find the description for x86 mwait:

Opcode	Instruction	Op/ En	64-Bit Mode	Compat/ Leg Mode	Description
0F 01 C9	MWAIT	ZO	Valid	Valid	A hint that allows the processor to stop instruction execution and enter an implementation-dependent optimized state until occurrence of a class of events.

From the table we can know MWAIT is an instruction optimized to support CPU C3/C6 state. Another document "Intel® 64 and IA-32 Architectures Software Developer's Manual" which is widely referenced by programmers also says: MONITOR and MWAIT are targeted for system software that supports efficient thread synchronization. So, during the Intel internal validation the platform test teams will not disable mwait in the multi-threading environment.

The customer disabled mwait to forbid the CPU enter into a C-state that is numerically higher than C1, try to get better latency performance, but immediately start running a multi-threaded test program. They do not realize that they have gave up part of the thread synchronization efficiency. This situation is similar to the illustration in Figure 1 (constraint 0), where the customer did something contradictory.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries.
Other names and brands may be claimed as the property of others.

We were very hard to reproduce this issue until finally realized the mwait has been turned off.

The similar scenario happens for the other customer discovered issues.

When we detect critical changes that are closely related to the issue, and with the appropriate workload, we can reproduce these issues in the laboratory and prevent the problem from leaking to customers.

2.5 How to avoid or reduce such risk in the future?

In addition to preventing known problems, is it possible for the BIOS developers to discover new issues in advance that may leak with the default BIOS configurations?

To research the possibility, the BIOS vendor needs a full picture for the customer BIOS changes at first, not only their default settings, but also the runtime configurations. This information should be available follow the workflow in figure 4 or figure 5 shows.

With the BIOS configuration change list, the items are too many that not all of them are related to the current issue, a sorting for the options is needed by their relevance to the issue.

At present, an engineering analysis method is used to obtain the result. The debug engineer maybe cannot root cause the issue very quickly, but at least they need to triage it to a specified domain, before the validation engineer begins to design the target case for the issue. By checking the domains between the modified options and the issue triage results, the debug engineers can give priority to attempt the BIOS options in the same domain. Sometimes, in a long time the debug engineers cannot reproduce the issue, before discovering a combination of options instead of a single key option.

When the options are sorted by their relevance to the issue, the verification engineers will attempt the options follow their relevance order. For a given option, the engineer can use an aggressive BIOS configuration strategy for experimental testing. The potential issue exposure rate should be different among the different settings selected by the BIOS vendor and customers separately, the aggressive BIOS configuration strategy requires the verification engineer to select the setting which will have higher exposure rate. The issue exposure rate is not obvious available, maybe the engineer need some rounds of experiments to get the result. With the accumulation of settings that follow this strategy, the exposure rate of platform issues should increase.

Figure 6 shows an example for this policy. There are 3 derivative BIOS with different configurations. B1 is the BIOS vendor release with default configurations, B2's configuration is changed follow the customers, BKC is the BIOS used for experiments, and it will be configured according to the aggressive strategy. The derivative B1 option F2 and F3 settings have higher exposure rate, while the derivative B2 have higher exposure rate configure for option F1, the platform validation team should select the F1 configuration from B2, and select the F2 and F3 option configurations from B1.

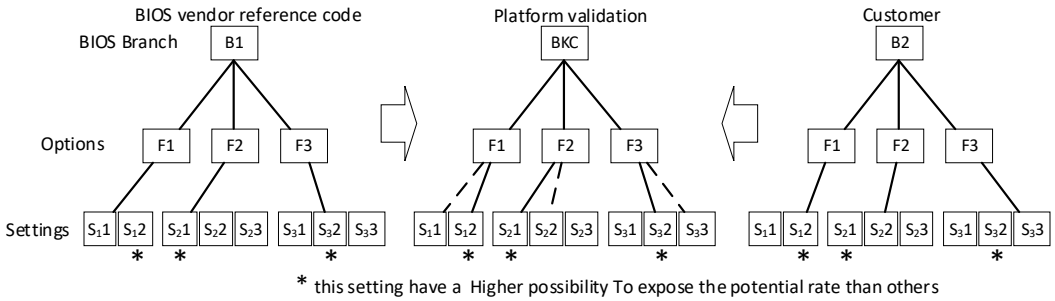

Figure 6: strategy for the options have known customer modifications

A regression comparison tests according to this strategy has been performed with 5 consecutive BKC (Best Known Configuration) external releases. These releases have been delivered to the customer at least more than 2 weeks, the customers will already report their issues if they have. The results show the aggressive BIOS configuration can discover more issues than the default configuration, it can discover some customer issues before the external release. With well-designed test cases the BIOS vendor can get an even better result.

Some simple regressions run as group A, the regressions run with 200 automation test cases on 5 consecutive BKC external releases. Compare with the BIOS vendor default settings, the platform BIOS configured follow the aggressive policy can discover 1 or 2 more issues.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

These results have some improvement but still do not good enough. Another round of regressions was run as group B, by keeping the test cases be updated by the ones new designed for the previous cycle external release test holes. The result is shown as table 1 below. The test results are normalized by subtract a certain value, until the BIOS configure 1 test results become 0.

	BIOS config 1	BIOS config 2	Ideal result
	Default config	Platform Validation config follow the policy in Figure 6	
Release 1	0	15	15
Release 2	0	9	9
Release 3	0	6	11
Release 4	0	8	12
Release 5	0	8	8

Table 1: Group B regression results (with aggressive BIOS configurations and new designed cases)

The column of ideal results refers to the total number of problems at the time of BIOS release. With the default BIOS configuration, it's impossible to reproduce some of the issues even with the new designed cases, because the necessary BIOS settings are not configured. Only with those options be changed, and run the related cases, the issues can be discovered.

Sometimes customers will run some new scenarios and may encounter problems at that time. No one has tried this new scenario before, including the software vendor. In this case, the BIOS vendor cannot discover the problem in advance of the customer.

3 Results

The customer BIOS default configuration modification is not only a technical problem, but also a customer requirement management problem. When they cannot get a BIOS with expected configuration the customer will go to an IBV (Independent BIOS Vendor), this behavior will cause some options of the IBV BIOS release to have a setting different from the original BIOS vendor.

For customers in a certain industry, the combination of BIOS options they need to modify is actually relatively fixed. With the effort to collect some BIOS option combinations on one platform, the BIOS vendor can reuse the results on the next generation platforms.

To a large extent, with the accumulated BIOS option tuning library, the platform validation team will has the ability to block the issues from being leakage to customers before release, by following the aggressive strategies, with the aggressive BIOS configuration policy and the well-designed new cases. The A/B regression test results in section 2.5 to verify the effectiveness of aggressive strategies are presented as the figure 7 below, 50% to 100% of the issues that were previously thought to be leaked to customers can be discovered before the BIOS release:

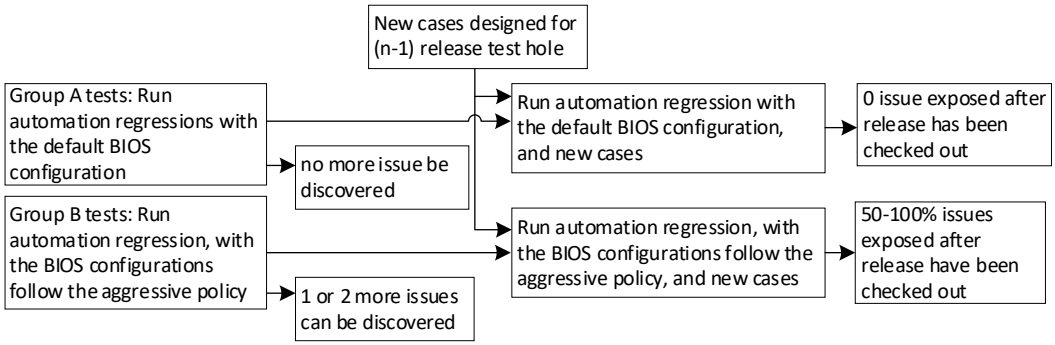

Figure 7: Regressions and the results

Every early discovered issue is important to help us accelerate the TTM. The active validation strategy with aggressive BIOS configuration and activity new case updating can help us discover new issues before leakage them to customers and is worth investing resources.

4 Summary

The customization for the BIOS default configuration exposes some new trends in customer demand management. This problem may not be unique to BIOS, the software products cannot satisfy all the main customers by a standard software configuration. The software vendors may need to clarify and update their

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

rules for the software default setting, it should be new feature oriented or customer usage oriented. If they select the later one, some communications with the customers or the independent software vendor are needed to collect their latest concerned demands. or even need to review who is the main customers update to date.

The communication with customer needs some improvement, too. During the research, we found some customer modifications are unnecessary, like the spectreV2 on Cooper Lake platform. The customers disabled spectreV2 patch based on their previous experience, they have an inherent impression that the system performance will greatly be impacted by this patch. But their impression is come from spectre software patch, while the spectreV2 is a hardware patch. The platform test result shows this option is nearly not impact the system performance, in all the monitored facts the worst performance degradation scale is about 1%. If we told customers this information and make sure they got the point, most of them will not change the default setting when unnecessary.

The aggressive validation strategies presented in this paper is not only applied for BIOS validation, but also applied for the existing server platform validation. The cooper lake platform passed the PV (Production Volume) milestone with no critical open issue at Q3 2020, that's the best score among the recent released platforms. These strategies will be applied to the next generation platforms and IPU (Intel Patch Update) validation in the future.

References

[1] Yoke Nicholas J, "Registers Used in UEFI Source", Jul 10, 2019. [Online]. Available: https://wiki.ith.intel.com/display/DcgUefiFw/Registers+Used+in+UEFI+Source

[2] "Intel® 64 and IA-32 Architectures Software Developer's Manual", May 2020. [Online]. Available: https://software.intel.com/content/dam/develop/external/us/en/documents-tps/325462-sdm-vol-1-2abcd-3abcd.pdf

[3] "Configuration management", 14 February 2021. [Online]. Available: https://en.wikipedia.org/wiki/Configuration_management

[4] "The Python Language Reference", for version 3.6, 26 Feb 2021. [Online]. Available: https://docs.python.org/3.6/reference/

[5] "The GNU C Reference Manual", [Online]. Available: https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html

[6] "pandas API reference — pandas 1.1.2 documentation", 8 September 2020. [Online]. Available: https://pandas.pydata.org/docs/reference/index.html

[7] "UEFI specification version 2.8", March 2019. [Online]. Available: https://uefi.org/sites/default/files/resources/UEFI_Spec_2_8_final.pdf

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.