MULTICS PLANNING NOTEBOOK

SYSTEM ANALYSIS AND PERFORMANCE IMPROVEMENT ACTIVITY

OVERALL GUIDELINES

Charles T. Clingen

1/26/68

Multics Planning Notebook

DRAFT 1/22/68

SYSTEM ANALYSIS AND PERFORMANCE IMPROVEMENT ACTIVITY -- OVERALL GUIDELINES

Corbato, Clingen

1. Introduction -- The Need for a New Activity

By the end of 1967, the MULTICS system had reached a sufficient level of completion that performance had become an important issue. The scope of the project was widening, as concepts were firmed and coordinated, to include the requirements of a productive system. It was discovered that unless system initialization times were significantly reduced, further integration of system modules would be limited to an unacceptably slow rate of progress. Performance was significantly improved and thus the MULTICS project entered a new phase in which performance became an increasingly pertinent and significant factor.

Subsequently, it has become increasingly apparent that an organized, orderly approach to the problem of performance improvement is needed. In response to this need, the System Analysis and Performance Improvement Activity has been established. Among the reasons for concentrating this work into a single Activity are:

- The concentration of performance-related knowledge and experience in one group reduces the dilution of knowledge and duplication of effort characteristic of uncoordinated, ad hoc performance improvement efforts.

- The

should result in improved efficiency of the people involved and improved response time for needed system changes.

.There will be a decrease in requirements for the time of MULTICS personnel best employed in other activities.

In summary, the urgency of a continuing performance improvement effort increases as the emphasis on MULTICS shifts from R + D of concepts toward system productivity. Also, there are compelling reasons for centralizing this effort into a single activity -- the System Analysis and Performance Improvement Activity.

2. MULTICS Characteristics Affecting Performance Improvement

There are at least three design and implementation attributes characteristic of MULTICS which are relevant to performance improvement:

.Because of the high content of new concepts and combinations of concepts in MULTICS, many areas of the system have a distinct R+D flavor with generality being introduced in order to facilitate later tuning as knowledge was gained rather than prematurely freezing the design. As a result of this generality, performance has suffered seriously.

.Although performance is becoming important even in currently running versions of the system, MULTICS is far from completely implemented and in some cases design remains to be done. Introduction of these as-yet unintegrated functions will without exception decrease system performance below the current level, in many cases drastically so, unless performance improvements take place on a continuing basis.

.MULTICS is coded in a high-level language. This language itself is new and as a result the language processor generates code the efficiency of which ranges from fair to extremely poor. This is

reflected as poor MULTICS performance.

Because of the above condition, the following broad strategies will be followed or recommended in each of the more specific tasks of individual module or subsystem improvement:

- Module or subsystem improvement will often be accomplished by selecting an adequate subset of the currently implemented logic such that unnecessary generality is removed without reducing functional capabilities offered. Often this is possible because design factors which are now frozen were still not fully understood at the time of initial implementation. Also, it may now be possible to recognize and combine nearly identical functions occurring in several unrelated areas into shared facilities, thereby improving performance and simplifying the design.
- Because the functional definition of the actual working system will be changing continually as new features are integrated, it will be difficult to relate performance levels of the various versions of MULTICS. Thus it is not unlikely that as new functions are added certain performance indicators will show degradation even as system improvement continues. Therefore, an attempt will be made to derive and utilize performance indicators independent, to the extent possible, of functional augmentation of the system.
- It will be possible to effect significant performance improvements in a function-independent manner by recoding algorithms so as to avoid EPL weaknesses. Further, recompilation of the system with improved versions of EPL and even selective recoding of MULTICS procedures in assembly language may occur and will result in significant improvement.

3. General Plan of Attack

In addition to the three above-mentioned strategies characteristic of all MULTICS performance improvement activities, the following more specific tactics will be followed for this Activity:

- The scope of the System Analysis and Performance Improvement Activity will not exceed those functions and modules to be included in Initial MULTICS.
- Primary attention will be devoted to improving the performance of the completely bootloaded system; only incidental attention will be given to bootloading efficiency.
- An "opportunistic" policy of identifying and analysing system "bottlenecks" will be adopted rather than selecting arbitrary modules for study. For example, quantitative analysis currently indicates that the major effect should be directed to analyzing and improving page fault, segment fault and linkage fault handling in that order.

Within these guidelines, the following generic steps or subsets thereof appear appropriate to the general analysis-improvement sequence for each area being analyzed.

- Study the design and implementation of the area of interest.
- Design the appropriate controlled experiments within a reproduceable environment to permit the measurement of the system performance factors of interest.
- Design and implement the software required to perform the experiment and process the results.
- Perform the experiments in the MULTICS environment.
- Reduce experimental data and distribute the results.

- Consult with designers of modules regarding significance of results.
- Select specific modules of functions for improvement.
- Recommend system modifications to improve performance to a specific predicted level.
- Upon receiving approval for modification from the Project, schedule the (re)design, (re)implementation and integration required for the improvement.
- Assist as appropriate in designing and implementing the system modification. The degree to which the System Analysis and Performance Improvement Activity should participate will be determined on a per-improvement basis and will range from recommendations and consultation to perhaps occasional reimplementation or test modifications.
- Measure performance of improved system to validate predicted performance increase.
- Recommend, if appropriate, integration of improvements into the standard system.

Scheduling and staffing of individual instances of these tasks as applied against specific functions or modules will be documented using the MULTICS Task Report.

In addition to these task-oriented steps general to the selection, measurement and analysis of each function being studied, there are several services which ideally should be developed and provided by the System Analysis and Performance Improvement Activity as a result of their experience. These include:

- The continued collection, analysis, and dissemination of EPL "lore" related to the performance pitfalls in EPL-generated

object code. Such information would be used to inform system programmers of the EPL language features to be used with care as well as to tell EPL compiler experts where improvements are most needed.

- Similarly, the collection and dissemination of information on the performance characteristics of MULTICS system primitives, allowing system programmers to make alternate choices of primitives where possible and permitting an intelligent ordering of primitives for re-implementation.
- The collection and dissemination of generic performance improvement techniques and criteria ranging from overall redesign suggestions for areas known to have caused trouble to data formats and EPL features to be used with care. The purpose of this information would be to guide system programmers the orderly upgrading of MULTICS functions.

The extent to which these "services" are provided depends largely upon the number and kinds of people in the activity.

4. Personnel Requirements

The System Analysis and Performance Activity is to be as nearly self-supporting as possible. This implies that a fairly wide diversity of talents and interests in the Activity. Among the types of activity expected are:

- Running of bootloads and dumps.
- Design and coding of MULTICS-compatible measurement procedures.
- Recommendations at the design, implementation, coding and EPL code production level regarding specific system modifications to be made by implementors responsible for affected modules.
- If appropriate, reimplementation or tentative test modifications of MULTICS modules.

Thus, personnel requirements seem to range from overall MULTICS understanding to EPLBSA coding and bootloading.

Initially, it seems that four to six people will be adequate to form the nucleus for this activity. By virtue of the size of the overall job of MULTICS performance improvement, the personnel in this activity must limit their activities primarily to analysis, measurement and recommendation, with actual implementation being done mostly by others.

5. Documentation

Documentation from the Activity will include:

- Task Report entries for specific tasks performed by this activity.
- MSPM sections if appropriate.
- Performance and analysis reports.
- Recommendations for system modifications if documentation is necessary.
- Record of performance improvements (If possible, a means should be determined for factoring out the effects of increasing system capability. At the least, a log or history of improvements will be maintained.)
- A per-procedure or per-module log or repository of suggestions for possible future improvements.
- A "manual" or checklist of possible items to be checked for the orderly performance upgrading of a generic module by a system programmer.

C.T. CLINGEN
1/26/68