SAS Programming Efficiency
A Brief Introduction

JC Wang

Western Michigan University
Department of Statistics
Outline

1 Introduction
   - Overview
   - Assessing Efficiency Needs
   - Benchmarking Resource Usages

2 Memory Usage
   - Controlling Memory Usage
   - Use of SASFILE Statement—An Introduction
Overview

If you write/maintain production programs and work with large data sets, it is important to optimize your programs with respect to the technical environment and the resource constraints at your site:

- CPU time
- real time (or user time or elapsed time)
- memory
- data storage space
- I/O
- development time (or programmer time)
Overview

If you write/maintain production programs and work with large data sets, it is important to optimize your programs with respect to the technical environment and the resource constraints at your site:

- CPU time
  - real time (or user time or elapsed time)
- memory
- data storage space
- I/O
- development time (or programmer time)
Overview

If you write/maintain production programs and work with large data sets, it is important to optimize your programs with respect to the technical environment and the resource constraints at your site:

- CPU time
- real time (or user time or elapsed time)
- memory
- data storage space
- I/O
- development time (or programmer time)
Overview

If you write/maintain production programs and work with large data sets, it is important to optimize your programs with respect to the technical environment and the resource constraints at your site:

- CPU time
- real time (or user time or elapsed time)
- memory
- data storage space
- I/O
- development time (or programmer time)
If you write/maintain production programs and work with large data sets, it is important to optimize your programs with respect to the technical environment and the resource constraints at your site:

- CPU time
- real time (or user time or elapsed time)
- memory
- data storage space
- I/O
- development time (or programmer time)
Overview

If you write/maintain production programs and work with large data sets, it is important to optimize your programs with respect to the technical environment and the resource constraints at your site:

- CPU time
- real time (or user time or elapsed time)
- memory
- data storage space
- I/O
- development time (or programmer time)
Overview

If you write/maintain production programs and work with large data sets, it is important to optimize your programs with respect to the technical environment and the resource constraints at your site:

- CPU time
- real time (or user time or elapsed time)
- memory
- data storage space
- I/O
- development time (or programmer time)
Threaded Processing
with SAS Scalable Performance Data Engine (version 9.1+)

- Threading improves performance by enabling a SAS session to use multiple I/O channels and multiple CPUs.
- A thread is a single, independent flow of control through a program or within a process.
- SAS Scalable Performance Data (SPD) Engine supports threaded I/O through partitioned data sets.
- Threaded enabled SAS procedures include: SORT, SQL, MEANS, SUMMARY, REPORT, and TABULATE.
Threaded Processing
with SAS Scalable Performance Data Engine (version 9.1+)

- Threading improves performance by enabling a SAS session to use multiple I/O channels and multiple CPUs.
- A thread is a single, independent flow of control through a program or within a process.
- SAS Scalable Performance Data (SPD) Engine supports threaded I/O through partitioned data sets.
- Threaded enabled SAS procedures include: SORT, SQL, MEANS, SUMMARY, REPORT, and TABULATE.
Threaded Processing
with SAS Scalable Performance Data Engine (version 9.1+)

- Threading improves performance by enabling a SAS session to use multiple I/O channels and multiple CPUs.
- A thread is a single, independent flow of control through a program or within a process.
- SAS Scalable Performance Data (SPD) Engine supports threaded I/O through partitioned data sets.
- Threaded enabled SAS procedures include: SORT, SQL, MEANS, SUMMARY, REPORT, and TABULATE.
Threaded Processing
with SAS Scalable Performance Data Engine (version 9.1+)

- Threading improves performance by enabling a SAS session to use multiple I/O channels and multiple CPUs.
- A thread is a single, independent flow of control through a program or within a process.
- SAS Scalable Performance Data (SPD) Engine supports threaded I/O through partitioned data sets.
- Threaded enabled SAS procedures include: SORT, SQL, MEANS, SUMMARY, REPORT, and TABULATE.
Threaded Processing
with SAS Scalable Performance Data Engine (version 9.1+)

- Threading improves performance by enabling a SAS session to use multiple I/O channels and multiple CPUs.
- A thread is a single, independent flow of control through a program or within a process.
- SAS Scalable Performance Data (SPD) Engine supports threaded I/O through partitioned data sets.
- Threaded enabled SAS procedures include: SORT, SQL, MEANS, SUMMARY, REPORT, and TABULATE.
Assessing Efficiency Needs
—Technical Environment

- Hardware: available memory, #CPUs, #/type of peripheral devices, communication hardware, network bandwidth, storage capacity, I/O bandwidth, capacity to upgrade
- Operating Environment: resource allocation, scheduling algorithms, I/O methods
- System Load: #users/jobs sharing system resources, network traffic, predicted increased load in the future
- SAS Environment: SAS products installed, #CPUs and memory allocated for SAS programming, methods running SAS
Assessing Efficiency Needs
—Technical Environment

- Hardware: available memory, #CPUs, #/type of peripheral devices, communication hardware, network bandwidth, storage capacity, I/O bandwidth, capacity to upgrade
- Operating Environment: resource allocation, scheduling algorithms, I/O methods
- System Load: #users/jobs sharing system resources, network traffic, predicted increased load in the future
- SAS Environment: SAS products installed, #CPUs and memory allocated for SAS programming, methods running SAS
Assessing Efficiency Needs
—Technical Environment

- **Hardware**: available memory, #CPUs, #/type of peripheral devices, communication hardware, network bandwidth, storage capacity, I/O bandwidth, capacity to upgrade
- **Operating Environment**: resource allocation, scheduling algorithms, I/O methods
  - System Load: #users/jobs sharing system resources, network traffic, predicted increased load in the future
  - SAS Environment: SAS products installed, #CPUs and memory allocated for SAS programming, methods running SAS
Assessing Efficiency Needs

—Technical Environment

- Hardware: available memory, #CPUs, #/type of peripheral devices, communication hardware, network bandwidth, storage capacity, I/O bandwidth, capacity to upgrade.

- Operating Environment: resource allocation, scheduling algorithms, I/O methods.

- System Load: #users/jobs sharing system resources, network traffic, predicted increased load in the future.

- SAS Environment: SAS products installed, #CPUs and memory allocated for SAS programming, methods running SAS.
Assessing Efficiency Needs
—Technical Environment

- **Hardware**: available memory, #CPUs, #/type of peripheral devices, communication hardware, network bandwidth, storage capacity, I/O bandwidth, capacity to upgrade
- **Operating Environment**: resource allocation, scheduling algorithms, I/O methods
- **System Load**: #users/jobs sharing system resources, network traffic, predicted increased load in the future
- **SAS Environment**: SAS products installed, #CPUs and memory allocated for SAS programming, methods running SAS
Assessing Efficiency Needs
—Programs

- Program size: focus on improving the efficiency of large programs.
- Times running the program: focus on improving programs that are run many times.
Assessing Efficiency Needs
—Programs

- Program size: focus on improving the efficiency of large program.
- Times running the program: focus on improving programs that are run many times.
Assessing Efficiency Needs
—Programs

- Program size: focus on improving the efficiency of large program.
- Times running the program: focus on improving programs that are run many times.
Assessing Efficiency Needs
—Data

- Data volume: focus on improving the efficiency of programs using large data sets or many data sets.
- Data type.
Assessing Efficiency Needs

—Data

- Data volume: focus on improving the efficiency of programs using large data sets or many data sets.
- Data type.
Assessing Efficiency Needs

—Data

- Data volume: focus on improving the efficiency of programs using large data sets or many data sets.
- Data type.
Assessing Efficiency Needs
—Understanding efficiency trade-offs

- disk space ↓ might ⇒ CPU time ↑
- I/O (reading/writing more data at one time) ↓ might ⇒ memory ↑
- real time (enabling threading) ↓ might ⇒ CPU time ↑
- any computing resource (increasing program complexity) ↓ might ⇒ programmer time ↑
Assessing Efficiency Needs
—Understanding efficiency trade-offs

- disk space ↓ might ➞ CPU time ↑
- I/O (reading/writing more data at one time) ↓ might ➞ memory ↑
- real time (enabling threading) ↓ might ➞ CPU time ↑
- any computing resource (increasing program complexity) ↓ might ➞ programmer time ↑
Assessing Efficiency Needs
—Understanding efficiency trade-offs

- disk space ↓ might ⇒ CPU time ↑
- I/O (reading/writing more data at one time) ↓ might ⇒ memory ↑
- real time (enabling threading) ↓ might ⇒ CPU time ↑
- any computing resource (increasing program complexity) ↓ might ⇒ programmer time ↑
Assessing Efficiency Needs
—Understanding efficiency trade-offs

- disk space ↓ might ⇒ CPU time ↑
- I/O (reading/writing more data at one time) ↓ might ⇒ memory ↑
- real time (enabling threading) ↓ might ⇒ CPU time ↑
- any computing resource (increasing program complexity) ↓ might ⇒ programmer time ↑
Assessing Efficiency Needs
—Understanding efficiency trade-offs

- disk space ↓ might ⇒ CPU time ↑
- I/O (reading/writing more data at one time) ↓ might ⇒ memory ↑
- real time (enabling threading) ↓ might ⇒ CPU time ↑
- any computing resource (increasing program complexity) ↓ might ⇒ programmer time ↑
Assessing Efficiency Needs
—Using SAS system options to track resources

1. OS: z/OS (z) or Unix/Linux (U) or Windows (W)
2. Option set: at invoking (i) or by OPTIONS statement (O)
3. Is default (d)

- STIMER (z/i/d, UW/iO/d): CPU time/real time to be tracked/written to SAS log
- MEMRPT (z/iO/d): memory usage statistics tracked
- FULLSTIMER (z/iO, UW/iO): all available resource usage statistics tracked/written to SAS log; alias (in z)=FULLSTATS and ignored if NOSTIMER and NOMEMRPT; some stats will not be accurate (in W) if this option is not (i)
- STATS (z/iO/d): write statistics that are tracked by any combination of options above
Assessing Efficiency Needs
—Using SAS system options to track resources

1. OS: z/OS (z) or Unix/Linux (U) or Windows (W)
2. Option set: at invoking (i) or by OPTIONS statement (O)
3. Is default (d)

- STIMER (z/i/d, UW/iO/d): CPU time/real time to be tracked/written to SAS log
- MEMRPT (z/iO/d): memory usage statistics tracked
- FULLSTIMER (z/iO, UW/iO): all available resource usage statistics tracked/written to SAS log; alias (in z)=FULLSTATS and ignored if NOSTIMER and NOMEMRPT; some stats will not be accurate (in W) if this option is not (i)
- STATS (z/iO/d): write statistics that are tracked by any combination of options above
Assessing Efficiency Needs
—Using SAS system options to track resources

1. OS: z/OS (z) or Unix/Linux (U) or Windows (W)
2. Option set: at invoking (i) or by OPTIONS statement (O)
3. Is default (d)

- **STIMER (z/i/d, UW/iO/d):** CPU time/real time to be tracked/written to SAS log
- **MEMRPT (z/iO/d):** memory usage statistics tracked
- **FULLSTIMER (z/iO, UW/iO):** all available resource usage statistics tracked/written to SAS log; alias (in z)=FULLSTATS and ignored if NOSTIMER and NOMEMRPT; some stats will not be accurate (in W) if this option is not (i)
- **STATS (z/iO/d):** write statistics that are tracked by any combination of options above
Assessing Efficiency Needs
—Using SAS system options to track resources

1. OS: z/OS (z) or Unix/Linux (U) or Windows (W)
2. Option set: at invoking (i) or by OPTIONS statement (O)
3. Is default (d)

- **STIMER (z/i/d, UW/iO/d):** CPU time/real time to be tracked/written to SAS log
- **MEMRPT (z/iO/d):** memory usage statistics tracked
- **FULLSTIMER (z/iO, UW/iO):** all available resource usage statistics tracked/written to SAS log; alias (in z)=FULLSTATS and ignored if NOSTIMER and NOMEMRPT; some stats will not be accurate (in W) if this option is not (i)
- **STATS (z/iO/d):** write statistics that are tracked by any combination of options above
Assessing Efficiency Needs
—Using SAS system options to track resources

1. OS: z/OS (z) or Unix/Linux (U) or Windows (W)
2. Option set: at invoking (i) or by OPTIONS statement (O)
3. Is default (d)

- STIMER (z/i/d, UW/iO/d): CPU time/real time to be tracked/written to SAS log
- MEMRPT (z/iO/d): memory usage statistics tracked
- FULLSTIMER (z/iO, UW/iO): all available resource usage statistics tracked/written to SAS log; alias (in z)=FULLSTATS and ignored if NOSTIMER and NOMEMRPT; some stats will not be accurate (in W) if this option is not (i)
- STATS (z/iO/d): write statistics that are tracked by any combination of options above
Benchmarking Resource Usages

- Turn on resource usage report options before benchmarking.
- Execute code for each programming technique in a separate SAS session and include only essential parts for performing the task.
- Run code for each programming technique several times.
- Run tests under conditions in which the production program will run.
- Turn off resource usage report options after benchmarking tests.
Bench marking Resource Usages

- Turn on resource usage report options before benchmarking.
- Execute code for each programming technique in a separate SAS session and include only essential parts for performing the task.
- Run code for each programming technique several times.
- Run tests under conditions in which the production program will run.
- Turn off resource usage report options after benchmarking tests.
Benchmarking Resource Usages

- Turn on resource usage report options before benchmarking.
- Execute code for each programming technique in a separate SAS session and include only essential parts for performing the task.
- Run code for each programming technique several times.
- Run tests under conditions in which the production program will run.
- Turn off resource usage report options after benchmarking tests.
Benchmarking Resource Usages

- Turn on resource usage report options before benchmarking.
- Execute code for each programming technique in a separate SAS session and include only essential parts for performing the task.
- Run code for each programming technique several times.
- Run tests under conditions in which the production program will run.
- Turn off resource usage report options after benchmarking tests.
Benchmarking Resource Usages

- Turn on resource usage report options before benchmarking.
- Execute code for each programming technique in a separate SAS session and include only essential parts for performing the task.
- Run code for each programming technique several times.
- **Run tests under conditions in which the production program will run.**
- Turn off resource usage report options after benchmarking tests.
Benchmarking Resource Usages

- Turn on resource usage report options before benchmarking.
- Execute code for each programming technique in a separate SAS session and include only essential parts for performing the task.
- Run code for each programming technique several times.
- Run tests under conditions in which the production program will run.
- Turn off resource usage report options after benchmarking tests.
Outline

1. Introduction
   - Overview
   - Assessing Efficiency Needs
   - Benchmarking Resource Usages

2. Memory Usage
   - Controlling Memory Usage
   - Use of SASFILE Statement—An Introduction
Measuring I/O

SAS DATA step

- inputs from SAS data (in storage device) to Buffer (memory), I/O measured here; then creates PDV (within memory); then transfers to Buffer (memory); then outputs to SAS data, I/O measured here.

- inputs from raw data (in storage device) to Buffer (memory), I/O measured here; then copies to input buffer (memory); then creates PDV (within memory); then transfers to Buffer (memory); then outputs to SAS data, I/O measured here.
Measuring I/O

SAS DATA step

- inputs from SAS data (in storage device) to Buffer (memory), I/O measured here; then creates PDV (within memory); then transfers to Buffer (memory); then outputs to SAS data, I/O measured here.

- inputs from raw data (in storage device) to Buffer (memory), I/O measured here; then copies to input buffer (memory); then creates PDV (within memory); then transfers to Buffer (memory); then outputs to SAS data, I/O measured here.
Measuring I/O

SAS DATA step

- inputs from SAS data (in storage device) to Buffer (memory), I/O measured here; then creates PDV (within memory); then transfers to Buffer (memory); then outputs to SAS data, I/O measured here.

- inputs from raw data (in storage device) to Buffer (memory), I/O measured here; then copies to input buffer (memory); then creates PDV (within memory); then transfers to Buffer (memory); then outputs to SAS data, I/O measured here.
Controlling Page Size and Number of Buffers

- **Page**: unit of data transfer between storage device and memory; includes #bytes used by the descriptor portion, the data values, and any overhead; is fixed in size when data set created, either to a default value or to a user-specified value.

- **Page size**: amount of data that can be transferred to one buffer in a single I/O

- Page size ↑ causes #times SAS read from/write to storage ↓ (at the cost of memory consumption ↑)

- Page size can be reported by PROC CONTENTS
Controlling Page Size and Number of Buffers

- Page: unit of data transfer between storage device and memory; includes #bytes used by the descriptor portion, the data values, and any overhead; is fixed in size when data set created, either to a default value or to a user-specified value.

- Page size: amount of data that can be transferred to one buffer in a single I/O

- page size ↑ causes #times SAS read from/write to storage ↓ (at the cost of memory consumption ↑)

- page size can be reported by PROC CONTENTS
Controlling Page Size and Number of Buffers

- **Page**: unit of data transfer between storage device and memory; includes bytes used by the descriptor portion, the data values, and any overhead; is fixed in size when data set created, either to a default value or to a user-specified value.

- **Page size**: amount of data that can be transferred to one buffer in a single I/O
  - page size ↑ causes #times SAS read from/write to storage ↓ (at the cost of memory consumption ↑)
  - page size can be reported by PROC CONTENTS
Controlling Page Size and Number of Buffers

- Page: unit of data transfer between storage device and memory; includes #bytes used by the descriptor portion, the data values, and any overhead; is fixed in size when data set created, either to a default value or to a user-specified value.

- Page size: amount of data that can be transferred to one buffer in a single I/O

- **Page size ↑ causes #times SAS read from/write to storage ↓ (at the cost of memory consumption ↑)**

- Page size can be reported by PROC CONTENTS
Controlling Page Size and Number of Buffers

- Page: unit of data transfer between storage device and memory; includes #bytes used by the descriptor portion, the data values, and any overhead; is fixed in size when data set created, either to a default value or to a user-specified value.

- Page size: amount of data that can be transferred to one buffer in a single I/O

- Page size ↑ causes #times SAS read from/write to storage ↓ (at the cost of memory consumption ↑)

- Page size can be reported by PROC CONTENTS
Controlling Page Size and Number of Buffers

continued

- Use `BUFSIZE=MIN|MAX|n` (in unit of bytes) option to control page size. Avoid `MIN`. Use `BUFSIZE=0` to reset to default.

- Use `BUFNO=MIN|MAX|n`. `MIN` (i.e., `n=0`) is the default, causing SAS to use the (OS-dependent) minimal optimal value. `MAX` is the (OS-dependent) maximum possible number which can reach up to $2^{31} - 1$. Recommended maximum is 10.

- The product of above numbers determines how much data can be transferred in one I/O operation.

- These two numbers have minimal effect on CPU usage.
Controlling Page Size and Number of Buffers

continued

- **Use** `BUFSIZE=MIN|MAX|n` (in unit of bytes) option to control page size. Avoid `MIN`. Use `BUFSIZE=0` to reset to default.

- **Use** `BUFNO=MIN|MAX|n`. `MIN` (i.e., `n=0`) is the default, causing SAS to use the (OS-dependent) minimal optimal value. `MAX` is the (OS-dependent) maximum possible number which can reach up to $2^{31} - 1$. Recommended maximum is 10.

- The product of above numbers determines how much data can be transferred in one I/O operation.

- These two numbers have minimal effect on CPU usage.
Controlling Page Size and Number of Buffers

Use \texttt{BUFSIZE=MIN|MAX|n} (in unit of bytes) option to control page size. Avoid \texttt{MIN}. Use \texttt{BUFSIZE=0} to reset to default.

Use \texttt{BUFNO=MIN|MAX|n}. \texttt{MIN} (i.e., \(n=0\)) is the default, causing SAS to use the (OS-dependent) minimal optimal value. \texttt{MAX} is the (OS-dependent) maximum possible number which can reach up to \(2^{31} - 1\). Recommended maximum is 10.

The product of above numbers determines how much data can be transferred in one I/O operation.

These two numbers have minimal effect on CPU usage.
Use `BUFSIZE=MIN | MAX | n` (in unit of bytes) option to control page size. Avoid `MIN`. Use `BUFSIZE=0` to reset to default.

Use `BUFNO=MIN | MAX | n`. `MIN` (i.e., `n=0`) is the default, causing SAS to use the (OS-dependent) minimal optimal value. `MAX` is the (OS-dependent) maximum possible number which can reach up to $2^{31} - 1$. Recommended maximum is 10.

The product of above numbers determines how much data can be transferred in one I/O operation.

These two numbers have minimal effect on CPU usage.
Controlling Page Size and Number of Buffers

continued

- Use `BUFSIZE=MIN | MAX | n` (in unit of bytes) option to control page size. Avoid `MIN`. Use `BUFSIZE=0` to reset to default.

- Use `BUFNO=MIN | MAX | n`. `MIN` (i.e., `n=0`) is the default, causing SAS to use the (OS-dependent) minimal optimal value. `MAX` is the (OS-dependent) maximum possible number which can reach up to $2^{31} - 1$. Recommended maximum is 10.

- The product of above numbers determines how much data can be transferred in one I/O operation.

- These two numbers have minimal effect on CPU usage.
Controlling Page Size and Number of Buffers

general recommendations

- For small data set, allocates as many buffers (BUFNO) as there are pages in the data set, especially when reading same observations several times during processing.

- Under z/OS, as data set size $\uparrow$, do: BUFNO $\uparrow$, don’t: BUFSIZE $\uparrow$. 
Controlling Page Size and Number of Buffers

general recommendations

- For small data set, allocates as many buffers ($\text{BUFNO}$) as there are pages in the data set, especially when reading same observations several times during processing.

- Under z/OS, as data set size ↑, do: $\text{BUFNO}$ ↑, don’t: $\text{BUFSIZE}$ ↑.
Controlling Page Size and Number of Buffers

general recommendations

- For small data set, allocates as many buffers (BUFNO) as there are pages in the data set, especially when reading same observations several times during processing.
- Under z/OS, as data set size ↑, do: BUFNO ↑, don’t: BUFSIZE ↑.
SASFILE Statement

- To hold an existing SAS data file in memory so that it is available to multiple program steps (for input or updating processing, but not for utility or output processing)
- To reduce I/O and CPU time in a program that reads multiple times the data.
- `SASFILE SAS-data<password-option(s)>` OPEN | LOAD | CLOSE

- I/O is reduced if there is sufficient *real* memory; otherwise, SAS might use virtual memory (resulting in degrading the performance) or might use the default # of buffers.
- If processing a part of a large SAS data repeatedly, use a DATA step with the SASFILE statement to create a subset of the data set that fits into memory.
SASFILE Statement

- To hold an existing SAS data file in memory so that it is available to multiple program steps (for input or updating processing, but not for utility or output processing)
- To reduce I/O and CPU time in a program that reads multiple times the data.

SASFILE SAS-data<password-option(s)>
OPEN | LOAD | CLOSE

- I/O is reduced if there is sufficient real memory; otherwise, SAS might use virtual memory (resulting in degrading the performance) or might use the default # of buffers.
- If processing a part of a large SAS data repeatedly, use a DATA step with the SASFILE statement to create a subset of the data set that fits into memory.
SASFILE Statement

- To hold an existing SAS data file in memory so that it is available to multiple program steps (for input or updating processing, but not for utility or output processing)
- To reduce I/O and CPU time in a program that reads multiple times the data.

SASFILE SAS-data<password-option(s)>
OPEN | LOAD | CLOSE

I/O is reduced if there is sufficient real memory; otherwise, SAS might use virtual memory (resulting in degrading the performance) or might use the default # of buffers.

If processing a part of a large SAS data repeatedly, use a DATA step with the SASFILE statement to create a subset of the data set that fits into memory.
SASFILE Statement

- To hold an existing SAS data file in memory so that it is available to multiple program steps (for input or updating processing, but not for utility or output processing)
- To reduce I/O and CPU time in a program that reads multiple times the data.

```
SASFILe SAS-data<password-option(s)> OPEN | LOAD | CLOSE
```

- I/O is reduced if there is sufficient real memory; otherwise, SAS might use virtual memory (resulting in degrading the performance) or might use the default # of buffers.
- If processing a part of a large SAS data repeatedly, use a DATA step with the SASFILE statement to create a subset of the data set that fits into memory.
SASFILE Statement

- To hold an existing SAS data file in memory so that it is available to multiple program steps (for input or updating processing, but not for utility or output processing)
- To reduce I/O and CPU time in a program that reads multiple times the data.

```
SASFILE SAS-data<password-option(s)> OPEN | LOAD | CLOSE
```

- I/O is reduced if there is sufficient *real* memory; otherwise, SAS might use virtual memory (resulting in degrading the performance) or might use the default # of buffers.

- If processing a part of a large SAS data repeatedly, use a DATA step with the SASFILE statement to create a subset of the data set that fits into memory.
SASFILE Statement

- To hold an existing SAS data file in memory so that it is available to multiple program steps (for input or updating processing, but not for utility or output processing)
- To reduce I/O and CPU time in a program that reads multiple times the data.

SASFILE  SAS-data<password-option(s)>
OPEN | LOAD | CLOSE

I/O is reduced if there is sufficient real memory; otherwise, SAS might use virtual memory (resulting in degrading the performance) or might use the default # of buffers.

If processing a part of a large SAS data repeatedly, use a DATA step with the SASFILE statement to create a subset of the data set that fits into memory.
IBUFSIZE= System option
for version 9+

1. Use IBUFSIZE=MIN | MAX | n system option to specify the page size of an index file, especially when
   - there are many levels in the index
   - index values are large in length

2. Avoid using MIN.

3. IBUFSIZE=0 resets to system default.
IBUFSTYPE= System option
for version 9+

1. use IBUFSTYPE=MIN | MAX | n system option to specify the page size of an index file, especially when
   - there are many levels in the index
   - index values are large in length

2. Avoid using MIN.

3. IBUFSTYPE=0 resets to system default.
IBUFSIZE= System option
for version 9+

1. use `IBUFSIZE=MIN | MAX | n` system option to specify the page size of an index file, especially when
   - there are many levels in the index
   - index values are large in length

2. Avoid using `MIN`.

3. `IBUFSIZE=0` resets to system default.
IBUFSIZE= System option
for version 9+

1. use \texttt{IBUFSIZE=MIN\mid MAX\mid n} system option to specify the page size of an index file, especially when
   - there are many levels in the index
   - index values are large in length

2. Avoid using \texttt{MIN}.

3. \texttt{IBUFSIZE=0} resets to system default.
IBUFSIZE= System option
for version 9+

1. use `IBUFSIZE=MIN | MAX | n` system option to specify the page size of an index file, especially when
   - there are many levels in the index
   - index values are large in length

2. Avoid using `MIN`.

3. `IBUFSIZE=0` resets to system default.
IBUFSIZE= System option for version 9+

1. use IBUFSIZE=MIN | MAX | n system option to specify the page size of an index file, especially when
   - there are many levels in the index
   - index values are large in length

2. Avoid using MIN.

3. IBUFSIZE=0 resets to system default.