基于对inetwork、icompute的日志治理经验，控制台输出日志过多，主要原因有以下几点：

1. icos ruquest id 在不停的打印。
2. 依赖其他的模块，不停打印与本模块无关的日志。
3. 同一个错误堆栈，在不同的位置重复多次打印。
4. 有些很明确的错误，只需要打印e.getMessage()一行错误信息即可，不需要打印完整错误堆栈。

基于以上原因分析，采用如下方案对两个模块的日志进行治理。

一、日志分类存储

1.1. 日志分类

1. 控制台日志
2. 全量日志
3. icos request id 日志
4. 与本模块无关日志

1.2. 日志分类存储配置示例

1. 控制台日志

```
<appender name="console" class="ch.qos.logback.core.ConsoleAppender">
    <!-- filter中定义的logger，日志不输出到控制台 -->
    <filter class="com.inspur.incloud.icompute.openstack.config.LogBackFilter"/>
    <encoder>
        <pattern>%d %p (%file:%line\)- %m%n</pattern>
        <!-- 控制台也要使用UTF-8，不要使用GBK，否则会中文乱码 -->
        <charset>UTF-8</charset>
    </encoder>
</appender>
```

LogBackFilter：用于定义哪些日志不输出到控制台

```
public class LogBackFilter extends Filter<ILoggingEvent> {

    // icos requestId相关日志
    private static final String [] REQUEST_LOGGER_NAMES = new String[] {
            "com.inspur.incloud.openstacksdk.connection.OpenstackRestClient"
    };

    // 与本模块无关日志logger
    private static final String [] CARELESS_LOGGER_NAMES = new String[] {
            "com.inspur.incloud.itask.asyntask.chain.impl.ChainTaskMonitorHelper",

 "springfox.documentation.swagger.readers.operation.OperationImplicitParameterReader",
```

```
            "com.inspur.incloud.itask.conf.XxlJobConfig",
            "org.openstack4j.connectors.httpclient.HttpResponseImpl"
    };

    @Override
    public FilterReply decide(ILoggingEvent iLoggingEvent) {
        final ArrayList<String> loggerNameList = new ArrayList<>();
        loggerNameList.addAll(Arrays.asList(REQUEST_LOGGER_NAMES));
        loggerNameList.addAll(Arrays.asList(CARELESS_LOGGER_NAMES));

        for (String loggerName : loggerNameList) {
            if (iLoggingEvent.getLoggerName().contains(loggerName)) {
                return FilterReply.DENY;
            }
        }

        return FilterReply.ACCEPT;
    }
}
```

2. 全量日志

```
<appender name="full" class="ch.qos.logback.core.rolling.RollingFileAppender">
    <File>${LOG_PATH}/${LOG_FILE}</File>
    <encoder>
        <pattern>%d %p (%file:%line\)- %m%n</pattern>
        <charset>UTF-8</charset>
    </encoder>
    <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
        <!-- 添加.gz 历史日志会启用压缩 大大缩小日志文件所占空间 -->
        <fileNamePattern>${LOG_PATH}/daily/${LOG_FILE}.%d{yyyy-MM-dd}.gz
        </fileNamePattern>
        <maxHistory>30</maxHistory><!-- 保留30天日志 -->
    </rollingPolicy>
</appender>
```

3. requestId相关日志

```
<appender name="icos-request-id" class="ch.qos.logback.core.rolling.RollingFileAppender">
    <File>${LOG_PATH}/icos-request-id.log</File>
    <encoder>
        <pattern>%d %p (%file:%line\)- %m%n</pattern>
        <charset>UTF-8</charset>
    </encoder>
    <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
        <fileNamePattern>${LOG_PATH}/daily/icos-request-id.log.%d{yyyy-MM-
dd}.gz</fileNamePattern>
        <maxHistory>30</maxHistory>
    </rollingPolicy>
</appender>
<logger name="com.inspur.incloud.openstacksdk.connection.OpenstackRestClient" level="INFO">
    <appender-ref ref="icos-request-id" />
</logger>
```

4. 与本模块无关日志

```
<appender name="careless" class="ch.qos.logback.core.rolling.RollingFileAppender">
    <File>${LOG_PATH}/careless.log</File>
    <encoder>
        <pattern>%d %p (%file:%line\)- %m%n</pattern>
        <charset>UTF-8</charset>
    </encoder>
    <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
        <fileNamePattern>${LOG_PATH}/daily/careless.log.%d{yyyy-MM-dd}.gz</fileNamePattern>
        <maxHistory>30</maxHistory>
    </rollingPolicy>
</appender>
<logger name="com.inspur.incloud.itask.asyntask.chain.impl.ChainTaskMonitorHelper"
level="INFO">
    <appender-ref ref="careless" />
</logger>
<logger
name="springfox.documentation.swagger.readers.operation.OperationImplicitParameterReader"
level="INFO">
    <appender-ref ref="careless" />
</logger>
<logger name="com.inspur.incloud.itask.conf.XxlJobConfig" level="INFO">
    <appender-ref ref="careless" />
</logger>
<logger name="org.openstack4j.connectors.httpclient.HttpResponseImpl" level="INFO">
    <appender-ref ref="careless" />
</logger>
```

二、处理不合理打印日志

2.1. 部分异常只需打印e.getMessage()

例如，不存在此资源异常，如浮动IP不存在、安全组不存在。通过错误码即可获取足够的信息，不需要打印完整的错误堆栈。以inetwork为例，该问题可在通用异常处理器中统一处理，示例代码如下：

```
@Slf4j
@ControllerAdvice
@ResponseBody
public class ControllerExceptionHandler {
    // 这些错误不打印详细日志
    final static List<String> exceptErrList = Arrays.asList("INETWORK_FLOATINGIP_NOT_EXIST"
            , "INETWORK_SECURITY_GROUP_NOT_EXIST", "INETWORK_VPC_NOT_EXIST",
"INETWORK_NET_NOT_EXIST");

    @ExceptionHandler(CloudBusinessException.class)
    public OperationResult<Object> handlerCloudBusinessException(CloudBusinessException e,
HttpServletRequest request) {
        // 1.打印错误日志
        log.error("uri ={}", request.getRequestURI());
        if (exceptErrList.contains(e.getMsgCode())) {
            log.error(e.getMessage());
        } else {
```

```
            log.error(e.getMessage(), e);
        }

        // 2.记录操作日志
        operLogUtils.writeOperLog(false, e);

        // 3.return
        return OperationResult.fail(e.getMsgCode(), e.getParamList());
    }

}
```

2.2. 部分异常不需要重复多次打印

例如，连接openstack资源池时，如果连接出错，会在不同位置重复打印四次日志。去除其中的三个，只保留一个在业务调用处的日志打印即可。

并且结合2.1章节，这些日志用有些只需要打印e.getMessage()，比如401问题、超时问题，这样可以进一步减少日志在控制台上的输出，以及在全量日志中的存储。

```
public class OpenstackConnPool {
    public static Object execute(Callable task, Long outTime, boolean needResponse) throws
RuntimeException {
        try {
           ...
        } catch (Exception e) {
            if (e.toString().contains("timed out")) {
                log.error(StringUtils.substring(e.toString(), 0, 100));
            } else {
                log.error(e.getMessage(), e);
            }
            throw new RuntimeException(e);
        }
        return null;
    }
}
```

```
public static OSClientObj connectCenter(OpenstackCenterConnectionInfo center, String projectId)
throws Exception {
        log.info("------ start connect openstack ceneter: " + center.getIp() + ", projectMor: "
+ projectId + " ------");
            OSClientV3 clientV3 = (OSClientV3) OpenstackConnPool.execute(new
OSClientThreadExtended(null, parameters) {
                @Override
                public OSClientV3 innerCall(OSClientV3 os, Object[] parameters) throws Exception
{
                    OSClientV3 osclientv3 = null;
                    try {
                        ...
                    } catch (RegionEndpointNotFoundException e) {
                        log.error(e.toString(), e);
                        throw e;
                    } catch (Exception e) {
                        if (e.toString().contains("timed out")) {
```

```
                        log.error(StringUtils.substring(e.toString(), 0, 100));
                    } else {
                        log.error(e.toString(), e);
                    }
                    throw e;
                }
                return osclientv3;
            }

        }, OpenstackConstant.THREAD_DEFAULT_WAIT_TIME);
        ...
    } catch (Exception e) {
        if (e.toString().contains("timed out")) {
            log.error(StringUtils.substring(e.toString(), 0, 100));
        } else {
            log.error(e.toString(), e);
        }
        throw e;
    }
    return osclientobj;
}
```

这三个地方，先改成只打印 e.getMessage()，后面确认完全没用再全部移除。

日志治理是一个持续的过程，需要持续的发现问题，解决问题。